



DeepLearning.AI

# Testing and Debugging

---

## Module introduction

# The discipline of software testing

- Manual testing
- Automated testing
- Performance testing
- Security testing



DeepLearning.AI

# Testing and Debugging

---

## Testing and debugging strategies



Image credit: NASA/JPL



```
from flask import Flask, jsonify
import threading

app = Flask(__name__)

@app.route('/api/greet/<name>', methods=['GET'])
def greet(name):
    return jsonify(message=f"Hello, {name}!")

if __name__ == '__main__':
    app.run(debug=True)
```

```
from flask import Flask, jsonify
import threading

app = Flask(__name__)

@app.route('/api/greet/<name>', methods=['GET'])
def greet(name):
    return jsonify(message=f"Hello, {name}!")

# To get it to work in Colab, use this line instead of the usual app.run
threading.Thread(target=app.run, kwargs={'host':'0.0.0.0','port':5000}).start()
```

```
from flask import Flask, jsonify
import threading

app = Flask(__name__)

@app.route('/api/greet/<name>', methods=['GET'])
def greet(name):
    return jsonify(message=f"Hello, {name}!")

# To get it to work in Colab, use this line instead of the usual app.run
threading.Thread(target=app.run, kwargs={'host':'0.0.0.0','port':5000}).start()
```

```
# When using the threading line above in Colab, you can now call your API in colab too!
import requests
req = requests.get("http://172.28.0.12:5000/api/greet/laurence")
print(req.status_code)
print(req.text)
```

```
from flask import Flask, jsonify
import threading

app = Flask(__name__)

@app.route('/api/greet/<name>', methods=['GET'])
def greet(name):
    return jsonify(message=f"Hello, {name}!")

# To get it to work in Colab, use this line instead of the usual app.run
threading.Thread(target=app.run, kwargs={'host':'0.0.0.0','port':5000}).start()
```

```
# When using the threading line above in Colab, you can now call your API in colab too!
import requests
req = requests.get("http://172.18.0.12:5000/api/greet/laurence")
print(req.status_code)
print(req.text)
```

# Prompting best practices

- **Be specific:** Provide detail and context about your problem
- **Assign a role:** Assign a role to tailor the output you receive
- **Request an expert opinion:** Assign an expert role and ask the LLM to evaluate the work you've already done to further refine it
- **Give feedback:** Iteratively prompt the LLM and provide feedback on the output you receive to get closer to your expected results

# Requesting tests

**Prompt** As an expert software tester who is teaching a new person how to write test cases, can you please analyze this code, and provide a set of test cases, explaining each one:

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/greet/<name>', methods=['GET'])
def greet(name):
    return jsonify(message=f"Hello, {name}!")

if __name__ == '__main__':
    app.run(debug=True)
```

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/greet/<name>', methods=['GET'])
def greet(name):
    return jsonify(message=f"Hello, {name}!")

if __name__ == '__main__':
    app.run(debug=True)
```

# Requesting tests

**Prompt** As an expert software tester who is teaching a new person how to write test cases, can you please analyze this code, and provide a set of test cases, explaining each one:

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/greet/<name>', methods=['GET'])
def greet(name):
    return jsonify(message=f"Hello, {name}!")

if __name__ == '__main__':
    app.run(debug=True)
```

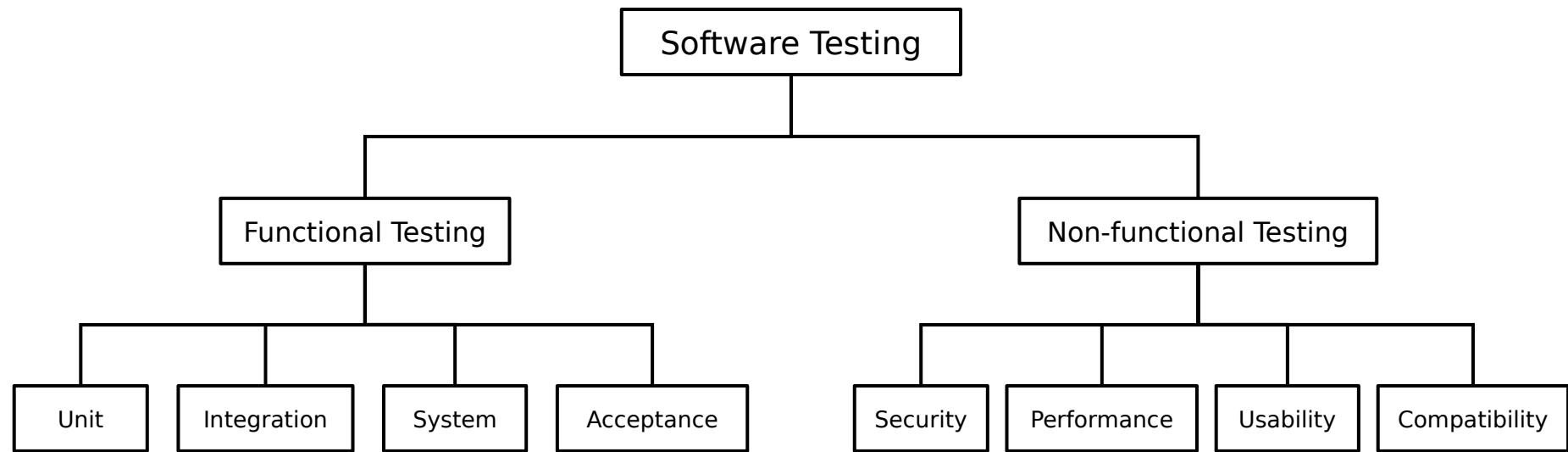


DeepLearning.AI

# Testing and Debugging

---

## Exploratory testing



```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added."

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found."

def list_tasks():
    return tasks
```

```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added."

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found."

def list_tasks():
    return tasks
```

```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added."

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found."

def list_tasks():
    return tasks

# Example usage
print(add_task("Buy groceries"))
print(add_task("Read a book"))
print(list_tasks())
print(remove_task("Read a book"))

print(list_tasks())
```

```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added."

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found."

def list_tasks():
    return tasks
```

```
print(remove_task("Go for a run")) # Output: Task not found.
```

```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added.

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found.

def list_tasks():
    return tasks
```

```
print(add_task("Buy groceries")) # Output: Task "Buy groceries" added.
print(list_tasks()) # Output: ['Buy groceries']
```

# Developing exploratory test cases

**Prompt** You are a software engineer and tester who is curious and who likes to go through code looking for edge cases. There's some Python code here -- please explore it and find any issues that might cause bugs or poor functionality:

```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added."

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found."

def list_tasks():
    return tasks
```

```
import unittest
```

```
class TestTodoList(unittest.TestCase):
```

```
    def setUp(self):
```

```
        global tasks
```

```
        tasks = []
```

```
    def test_add_task(self):
```

```
        self.assertEqual(add_task("Buy groceries"), "Task 'Buy groceries' added.")
```

```
        self.assertIn("Buy groceries", tasks)
```

```
    def test_remove_task(self):
```

```
        add_task("Read a book")
```

```
        self.assertEqual(remove_task("Read a book"), "Task 'Read a book' removed.")
```

```
        self.assertNotIn("Read a book", tasks)
```

```
def test_remove_nonexistent_task(self):
    self.assertEqual(remove_task("Go for a run"), "Task not found.")

def test_list_tasks(self):
    add_task("Buy groceries")
    add_task("Read a book")
    self.assertEqual(list_tasks(), ["Buy groceries", "Read a book"])

def test_add_empty_task(self):
    self.assertEqual(add_task(""), "Task '' added.")
    self.assertIn("", tasks)

if __name__ == '__main__':
    unittest.main()
```

```
import unittest
```

```
class TestTodoList(unittest.TestCase):
```

```
    def setUp(self):
        global tasks
        tasks = []
```

```
    def test_add_task(self):
```

```
        self.assertEqual(add_task("Buy groceries"), "Task 'Buy groceries' added.")
        self.assertIn("Buy groceries", tasks)
```

```
    def test_remove_task(self):
```

```
        add_task("Read a book")
        self.assertEqual(remove_task("Read a book"), "Task 'Read a book' removed.")
        self.assertNotIn("Read a book", tasks)
```

```
import unittest
```

```
class TestTodoList(unittest.TestCase):
```

```
    def setUp(self):
```

```
        global tasks
```

```
        tasks = []
```

```
    def test_add_task(self):
```

```
        self.assertEqual(add_task("Buy groceries"), "Task 'Buy groceries' added.")
```

```
        self.assertIn("Buy groceries", tasks)
```

```
    def test_remove_task(self):
```

```
        add_task("Read a book")
```

```
        self.assertEqual(remove_task("Read a book"), "Task 'Read a book' removed.")
```

```
        self.assertNotIn("Read a book", tasks)
```

```
def test_remove_nonexistent_task(self):
    self.assertEqual(remove_task("Go for a run"), "Task not found.")

def test_list_tasks(self):
    add_task("Buy groceries")
    add_task("Read a book")
    self.assertEqual(list_tasks(), ["Buy groceries", "Read a book"])

def test_add_empty_task(self):
    self.assertEqual(add_task(""), "Task '' added.")
    self.assertIn("", tasks)

if __name__ == '__main__':
    unittest.main()
```



DeepLearning.AI

# Testing and Debugging

---

## Functional testing

```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added."

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found."

def list_tasks():
    return tasks

# Example usage
print(add_task("Buy groceries"))
print(add_task("Read a book"))
print(list_tasks())
print(remove_task("Read a book"))
print(list_tasks())
print(add_task("")) # Output: Task '' added.
print(list_tasks()) # Output: ['Buy groceries', '']
```

# Ask an LLM to create functional tests (2)

**Prompt** I'm working on a simple todo list app. Here is the code:

```
.. - ..  
tasks = []  
  
def add_task(task):  
    tasks.append(task)  
    return f"Task '{task}' added."  
  
def remove_task(task):  
    if task in tasks:  
        tasks.remove(task)  
        return f"Task '{task}' removed."  
    else:  
        return "Task not found."  
  
def list_tasks():  
    return tasks
```

**Prompt** As an expert software tester, write code that converts the output of the

**Prompt** ~~exploratory testing to different cases with the list of module in python.~~



```
import unittest
```

```
class TestTodoList(unittest.TestCase):
```

```
    def setUp(self):
```

```
        global tasks
```

```
        tasks = []
```

```
    def test_add_task(self):
```

```
        self.assertEqual(add_task("Buy groceries"), "Task 'Buy groceries' added.")
```

```
        self.assertIn("Buy groceries", tasks)
```

```
    def test_remove_task(self):
```

```
        add_task("Read a book")
```

```
        self.assertEqual(remove_task("Read a book"), "Task 'Read a book' removed.")
```

```
        self.assertNotIn("Read a book", tasks)
```

```
def test_remove_nonexistent_task(self):
    self.assertEqual(remove_task("Go for a run"), "Task not found.")

def test_list_tasks(self):
    add_task("Buy groceries")
    add_task("Read a book")
    self.assertEqual(list_tasks(), ["Buy groceries", "Read a book"])

def test_add_empty_task(self):
    self.assertEqual(add_task(""), "Task '' added.")
    self.assertIn("", tasks)

if __name__ == '__main__':
    unittest.main()
```

```
import unittest
```

```
class TestTodoList(unittest.TestCase):
```

```
    def setUp(self):
        global tasks
        tasks = []
```

```
    def test_add_task(self):
```

```
        self.assertEqual(add_task("Buy groceries"), "Task 'Buy groceries' added.")
```

```
        self.assertIn("Buy groceries", tasks)
```

```
    def test_remove_task(self):
```

```
        add_task("Read a book")
```

```
        self.assertEqual(remove_task("Read a book"), "Task 'Read a book' removed.")
```

```
        self.assertNotIn("Read a book", tasks)
```

```
import unittest
```

```
class TestTodoList(unittest.TestCase):
```

```
    def setUp(self):
```

```
        global tasks
```

```
        tasks = []
```

```
    def test_add_task(self):
```

```
        self.assertEqual(add_task("Buy groceries"), "Task 'Buy groceries' added.")
```

```
        self.assertIn("Buy groceries", tasks)
```

```
    def test_remove_task(self):
```

```
        add_task("Read a book")
```

```
        self.assertEqual(remove_task("Read a book"), "Task 'Read a book' removed.")
```

```
        self.assertNotIn("Read a book", tasks)
```

```
def test_remove_nonexistent_task(self):
    self.assertEqual(remove_task("Go for a run"), "Task not found.")

def test_list_tasks(self):
    add_task("Buy groceries")
    add_task("Read a book")
    self.assertEqual(list_tasks(), ["Buy groceries", "Read a book"])

def test_add_empty_task(self):
    self.assertEqual(add_task(""), "Task '' added.")
    self.assertIn("", tasks)

if __name__ == '__main__':
    unittest.main()
```

```
tasks = []

def add_task(task):
    tasks.append(task)
    return f"Task '{task}' added."

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return f"Task '{task}' removed."
    else:
        return "Task not found."

def list_tasks():
    return tasks
```

```
def test_remove_nonexistent_task(self):
    self.assertEqual(remove_task("Go for a run"), "Task not found.")

def test_list_tasks(self):
    add_task("Buy groceries")
    add_task("Read a book")
    self.assertEqual(list_tasks(), ["Buy groceries", "Read a book"])
```

```
def test_add_empty_task(self):
    self.assertEqual(add_task(""), "Task '' added.")
    self.assertIn("", tasks)
```

```
if __name__ == '__main__':
    unittest.main()
```

# Write new test cases for updated code

**Prompt** As an expert software tester, write code that converts the output of the exploratory testing into functional tests using the unittest module in python.



DeepLearning.AI

# Testing and Debugging

---

## Automated testing



Manual testing



Automated testing



Q Search

## About pytest

pytest is a mature full-featured Python testing tool that helps you write better programs.

## Contents

- [Home](#)
- [Get started](#)
- [How-to guides](#)
- [Reference guides](#)
- [Explanation](#)
- [Complete table of contents](#)
- [Library of examples](#)

## About the project

Changelog

# pytest: helps you write better programs

The `pytest` framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

`pytest` requires: Python 3.8+ or PyPy3.

PyPI package name: [pytest](#)

## A quick example

```
# content of test_sample.py
def inc(x):
    return x + 1

def test_answer():
    assert inc(3) == 5
```

### Next Open Trainings and Events

[pytest development sprint, June 17th – 22nd 2024](#), Klaus (AT) / Remote

[pytest tips and tricks for a better testsuite, at Europython 2024](#), July 8th – 14th 2024 (3h), Prague (CZ)

[pytest: Professionelles Testen \(nicht nur für Python, at CH Open Workshoptage, September 2nd 2024](#), HSLU Rotkreuz (CH)

[Professional Testing with Python, via Python Academy](#) (3 day in-depth training), March 4th – 6th 2025, Leipzig (DE) / Remote

Also see [previous talks and blogposts](#)



```
tasks = []

def add_task(task):
    if not task:
        raise ValueError("Task cannot be empty.")
    tasks.append(task)
    return tasks

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return tasks
    else:
        return "Task not found."

def list_tasks():
    return tasks

def clear_tasks():
    tasks.clear()

    return "Tasks cleared."
```

```
pip install pytest
```

```
✓ todolist project
  ↘ code
    tasks.py
    test_todoList.py
> other project
```

Find tests that start with “test\_”  
in your code

Find tests in separate files  
that start “test\_”

```
import pytest

def test_add_task():
    tasks = []
    assert add_task('Buy groceries') == ['Buy groceries']
    assert add_task('Read a book') == ['Buy groceries', 'Read a book']

def test_list_tasks():
    clear_tasks()
    assert list_tasks() == []
    add_task('Buy groceries')
    assert list_tasks() == ['Buy groceries']
    add_task('Go for a run')

    assert list_tasks() == ['Buy groceries', 'Go for a run']
```

```
===== test session starts =====
platform linux -- Python 3.10.12, pytest-7.4.4, pluggy-1.5.0
rootdir: /content
plugins: anyio-3.7.1
collected 2 items

tasks.py F. [100%]

===== FAILURES =====
____ test_add_task ____

def test_add_task():
    tasks = []
>     assert add_task('Buy groceries') == ['Buy groveries']
E     AssertionError: assert ['Buy groceries'] == ['Buy groveries']
E         At index 0 diff: 'Buy groceries' != 'Buy groveries'
E         Use -v to get more diff

tasks.py:24: AssertionError
===== short test summary info =====
FAILED tasks.py::test_add_task - AssertionError: assert ['Buy groceries'] == ['Buy groveries']
===== 1 failed, 1 passed in 0.10s =====
```

```
===== test session starts =====
platform linux -- Python 3.10.12, pytest-7.4.4, pluggy-1.5.0
rootdir: /content
plugins: anyio-3.7.1
collected 2 items

tasks.py F. [100%]

===== FAILURES =====
____ test_add_task ____

def test_add_task():
    tasks = []
>     assert add_task('Buy groceries') == ['Buy groveries']
E     AssertionError: assert ['Buy groceries'] == ['Buy groveries']
E         At index 0 diff: 'Buy groceries' != 'Buy groveries'
E         Use -v to get more diff

tasks.py:24: AssertionError
===== short test summary info =====
FAILED tasks.py::test_add_task - AssertionError: assert ['Buy groceries'] == ['Buy groveries']
===== 1 failed, 1 passed in 0.10s =====
```

# Generating tests

**Prompt:** You are an expert in PyTest for automated testing of python code. Please amend this code with a comprehensive set of tests in Pytest to find bugs or other issues in the code.

```
tasks = []

def add_task(task):
    if not task:
        raise ValueError("Task cannot be empty.")
    tasks.append(task)
    return tasks

def remove_task(task):
    if task in tasks:
        tasks.remove(task)
        return tasks
    else:
        return "Task not found."

def list_tasks():
    return tasks

def clear_tasks():
    tasks.clear()
    return "Tasks cleared."
```



DeepLearning.AI

# Testing and Debugging

---

## Software performance testing

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
```

```
def sum_of_primes_naive(numbers):
    total = 0
    for number in numbers:
        if is_prime(number):
            total += number
    return total
```

# Performance testing a function

**Prompt** How would I measure how long it takes to call the sum\_of\_primes\_naive function in Python?

```
def sum_of_primes_naive(numbers):
    total = 0
    for number in numbers:
        if is_prime(number):
            total += number

    return total
```



ChatGPT

You can measure the execution time of the `sum\_of\_primes\_naive` function in Python using the `time`, `timeit`, and `cProfile` modules. Here are three methods to achieve this:

## Method 1: Using `time` Module

This method measures the wall-clock time before and after the function call.

python

Copy code

```
import time

# Sample list of numbers
numbers = list(range(1, 10000))

# Measure time
start_time = time.time()
sum_of_primes_naive(numbers)
end_time = time.time()
```

```
# Measure time using timeit
import timeit

execution_time = timeit.timeit('sum_of_primes_naive(numbers)',
                               globals=globals(), number=1)
print(f"Execution time: {execution_time} seconds")
```

✓ 41s

```
[10] # Generate a list of numbers  
numbers = list(range(1, 100000))  
sum_of_primes_naive(numbers)
```

→ 454396537

✓ 40s

```
# Measure time using timeit  
execution_time = timeit.timeit('sum_of_primes_naive(numbers)', globals=globals(), number=1)  
print(f"Execution time: {execution_time} seconds")
```

→ Execution time: 40.20760166499997 seconds

```
# Measure time using timeit
import timeit

execution_time = timeit.timeit('sum_of_primes_naive(numbers)',  
                               globals=globals(), number=1)  
  
print(f"Execution time: {execution_time} seconds")
```

```
import cProfile  
  
cProfile.run('sum_of_primes_naive(numbers)')
```

2s

```
cProfile.run('sum_of_primes_naive(numbers)')
```



100003 function calls in 52.203 seconds

Ordered by: standard name

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.040	0.040	52.203	52.203	<ipython-input-3-ecbf73245c3a>:12(sum_of_primes_naive)
99999	52.156	0.001	52.156	0.001	<ipython-input-3-ecbf73245c3a>:4(is_prime)
1	0.000	0.000	52.203	52.203	<string>:1(<module>)
1	0.000	0.000	52.203	52.203	{built-in method builtins.exec}
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}

# Using cProfile output in your prompt

**Prompt :** Rewrite my is\_prime function in the code below to be more optimized. Below the code is the output from a CProfile analysis of the function.

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

def sum_of_primes_naive(numbers):
    total = 0
    for number in numbers:
        if is_prime(number):
            total += number
    return total
```

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.046	0.046	52.203	52.203	<ipython-input-3-ecbf73245c3a>:12(sum_of_primes_naive)
99999	52.156	0.001	52.156	0.001	<ipython-input-3-ecbf73245c3a>:4(is_prime)
1	0.000	0.000	52.203	52.203	<string>:1(<module>)
1	0.000	0.000	52.203	52.203	{built-in method builtins.exec}
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}

```
def is_prime_optimized(n):  
    if n <= 1:  
        return False  
    if n <= 3:  
        return True  
    if n % 2 == 0 or n % 3 == 0:  
        return False  
    i = 5  
    while i * i <= n:  
        if n % i == 0 or n % (i + 2) == 0:  
            return False  
        i += 6  
    return True
```

→ Optimized Implementation: Sum of primes = 454396537, Time taken = 0.11378026008605957 seconds

[17]: get\_user\_time = timeit.timeit(lambda: sum\_of\_primes\_optimized(numbers), number=1)  
print(f"Execution time for sum\_of\_primes: {get\_user\_time} seconds")

→ Execution time for sum\_of\_primes: 0.13188440499993703 seconds

# Prompting best practices

- **Be specific:** Provide detail and context about your problem
- **Assign a role:** Assign a role to tailor the output you receive
- **Request an expert opinion:** Assign an expert role and ask the LLM to evaluate the work you've already done to further refine it
- **Give feedback:** Iteratively prompt the LLM and provide feedback on the output you receive to get closer to your expected results

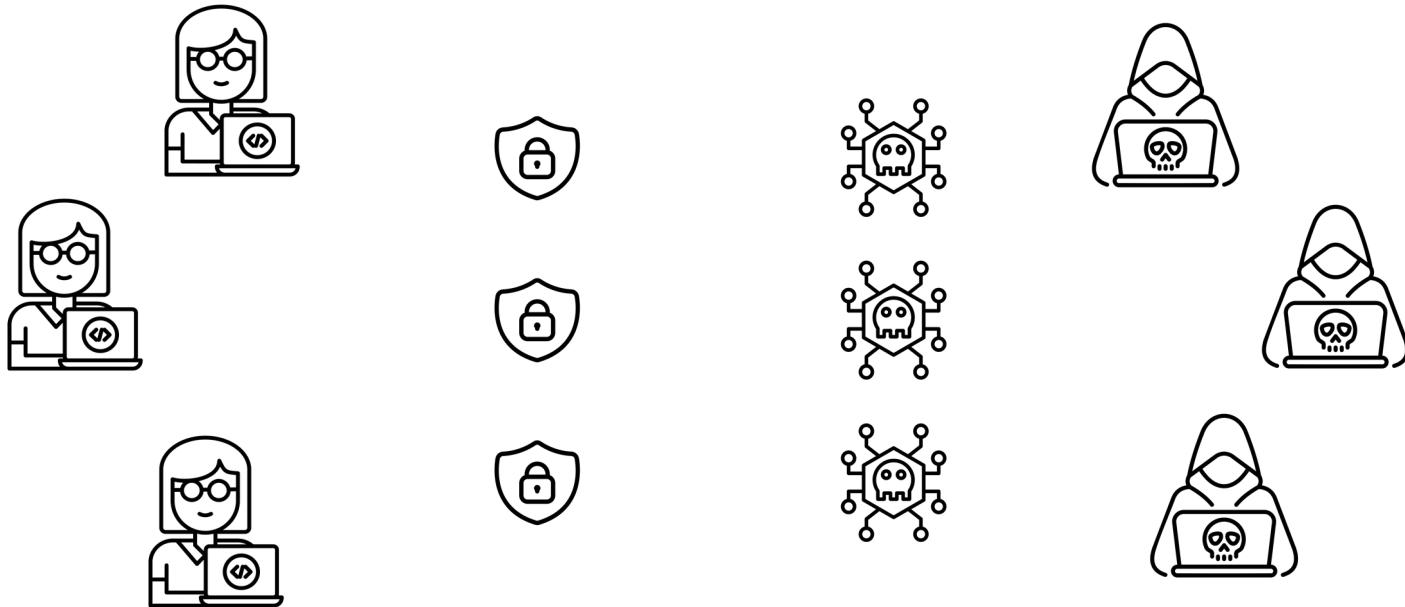


DeepLearning.AI

# Testing and Debugging

---

## Security testing



Search

⌘ K

## GET STARTED

Overview

Quickstart

Concepts

## Models

Overview

Model updates

## GPT-4o

GPT-4o mini

GPT-4 Turbo and GPT-4

GPT-3.5 Turbo

DALL-E

TTS

Whisper

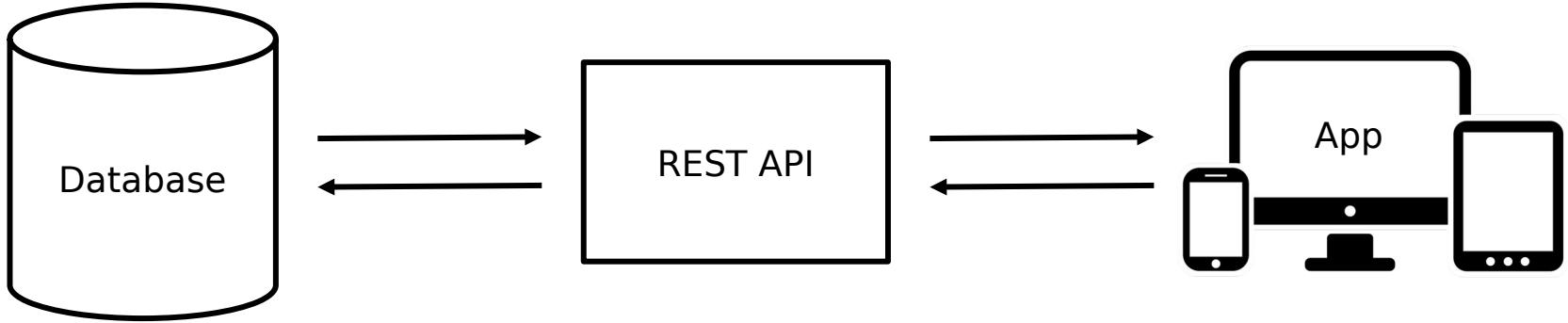
Embeddings

Moderation

## GPT-4o

GPT-4o (“o” for “omni”) is our most advanced model. It is multimodal (accepting text or image inputs and outputting text), and it has the same high intelligence as GPT-4 Turbo but is much more efficient—it generates text 2x faster and is 50% cheaper. Additionally, GPT-4o has the best vision and performance across non-English languages of any of our models. GPT-4o is available in the OpenAI API to paying customers. Learn how to use GPT-4o in our [text generation guide](#).

MODEL	DESCRIPTION	CONTEXT WINDOW	MAX OUTPUT TOKENS	TRAINING DATA
gpt-4o	<p><b>GPT-4o:</b> Our high-intelligence flagship model for complex, multi-step tasks. GPT-4o is cheaper and faster than GPT-4 Turbo.</p> <p>Currently points to <a href="#">gpt-4o-2024-05-13</a> [1].</p>	128,000 tokens	4,096 tokens	Up to Oct 2023
gpt-4o-2024-05-13	gpt-4o currently points to this version.	128,000 tokens	4,096 tokens	Up to Oct 2023
gpt-4o-2024-08-06	Latest snapshot that supports <a href="#">Structured Outputs</a>	128,000 tokens	16,384 tokens	Up to Oct 2023
chatgpt-4o-latest	Dynamic model continuously updated to the current version of GPT-4o in ChatGPT. <a href="#">Intended for research and evaluation</a> [2].	128,000 tokens	16,384 tokens	Up to Oct 2023



```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)
```

```
@app.route('/user/<int:id>', methods=['GET'])

def get_user(id):
    user = User.query.get(id)
    if user:
        return jsonify({"id": user.id,
                       "username": user.username,
                       "password": user.password})
    return jsonify({"message": "User not found"}), 404
```

`http://<yourservice>/user/1`

`http://<yourservice>/user/1`

`SELECT * from USERS where ID=1`

`http://<yourservice>/user/1'DROP TABLE  
USERS`

`http://<yourservice>/user/1'DROP TABLE  
USERS`

`SELECT * FROM USERS WHERE ID=1' DROP  
TABLE USERS`

# Prompting best practices

- **Be specific:** Provide detail and context about your problem
- **Assign a role:** Assign a role to tailor the output you receive
- **Request an expert opinion:** Assign an expert role and ask the LLM to evaluate the work you've already done to further refine it
- **Give feedback:** Iteratively prompt the LLM and provide feedback on the output you receive to get closer to your expected results

# Generating security tests

**Prompt** You are an expert in web security and in creating API endpoints. With the following code, there are likely many vulnerabilities. Can you create some test cases that test against these vulnerabilities?

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)

@app.route('/user/<int:id>', methods=['GET'])
def get_user(id):
    user = User.query.get(id)
    if user:
        return jsonify({ "id": user.id,
                        "username": user.username,
                        "password": user.password})
    return jsonify({ "message": "User not found"}), 404
```

1. SQL Injection: The application might be vulnerable to SQL injection if it does not properly sanitize inputs.
2. Cross-Site Scripting (XSS): The application might be vulnerable to XSS attacks if it does not properly sanitize inputs.
3. Insecure Password Storage: Passwords are stored in plain text, which is insecure.
4. Broken Authentication and Session Management: There is no authentication or session management implemented.
5. Insufficient Authorization: No checks are performed to ensure the requester has permission to perform actions on resources.
6. Insecure Direct Object References: Direct access to user records by ID without authorization checks.
7. Data Exposure: Sensitive information (passwords) is returned in API responses.

<https://flask.palletsprojects.com/en/3.0.x/>

## Project Links

[Donate](#)  
[PyPI Releases](#)  
[Source Code](#)  
[Issue Tracker](#)  
[Chat](#)

## Contents

[Welcome to Flask](#)  
[User's Guide](#)  
[API Reference](#)  
[Additional Notes](#)

## Quick search

# Flask

Welcome to Flask's documentation. Get started with [Installation](#) and then get an overview with the [Quickstart](#). There is also a more detailed [Tutorial](#) that shows how to create a small but complete application with Flask. Common patterns are described in the [Patterns for Flask](#) section. The rest of the docs describe each component of Flask in detail, with a full reference in the [API](#) section.

Flask depends on the [Werkzeug](#) WSGI toolkit, the [Jinja](#) template engine, and the [Click](#) CLI toolkit. Be sure to check their documentation as well as Flask's when looking for information.

## User's Guide

Flask provides configuration and conventions, with sensible defaults, to get started. This section of the documentation explains the different parts of the Flask framework and how they can be used, customized, and extended. Beyond Flask itself, look for community-maintained extensions to add even more functionality.

Step 1: Create and run test cases for potential security issues

Step 1: Create and run test cases for potential security issues

Step 2: Improve your code to mitigate issues found in testing

Step 1: Create and run test cases for potential security issues



Step 2: Improve your code to mitigate issues found in testing

REPEAT!



DeepLearning.AI

# Testing and Debugging

---

Flask example  
walkthrough (Optional)



DeepLearning.AI

# Documentation

---

## Module introduction

```
function calcFact() {  
    let n = prompt("Enter a number:");  
    if (n < 0) {  
        alert("Invalid input!");  
        return;  
    }  
    let r = 1;  
    let i = 1;  
    (function f() {  
        if (i <= n) {  
            r *= i;  
            i++;  
            setTimeout(f, 0);  
        } else {  
            alert("Factorial of " + n + " is " + r);  
        }  
    })();  
}  
  
calcFact();
```

```
function af() {  
    let x = 10;  
    let y = 20;
```

-- mu-recur

fact :: Integer -> Integer

fact = fix (\f n -> if n == 0 then 1 else n \* f (n - 1))

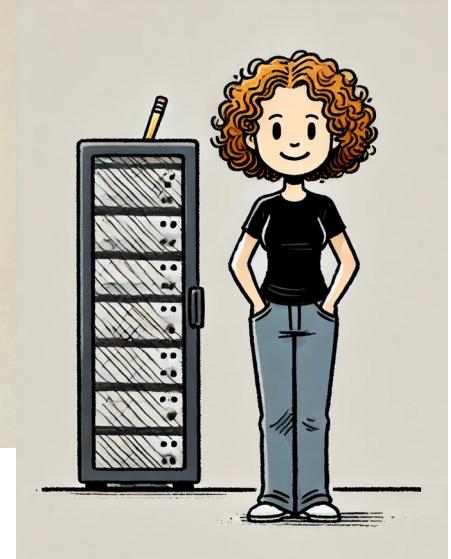
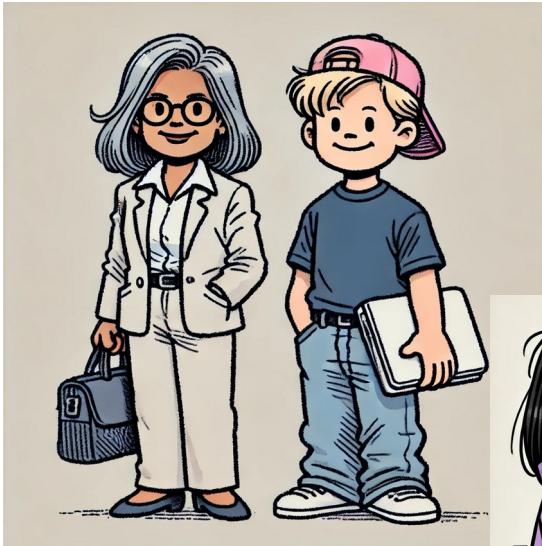
```
def calculate_square_area(side_length):
    """
    Calculate the area of a square.

    Parameters:
    side_length (float): The length of a side of the square.

    Returns:
    float: The area of the square.
    """
    # Check if the side length is a positive number
    if side_length <= 0:
        raise ValueError("Side length must be a positive number")

    # Calculate the area by squaring the side length
    area = side_length ** 2 # ** is the square function!

    # Return the calculated area
    return area
```





DeepLearning.AI

# Documentation

---

Principles of good  
documentation

```
# Initialize x with the value 10
```

```
x = 10
```

```
def add(a, b):
```

```
    """
```

Add two numbers and return the result.

Parameters:

a (int, float): The first number.

b (int, float): The second number.

Returns:

int, float: The sum of the two numbers.

```
    """
```

```
return a + b
```

# Benefits of good code documentation

- Improves code readability
- Prevents technical debt
- Helps others learn to use your code
- Increases overall code quality

# Principles of good documentation writing

- Be clear and concise

...P ----- P -- P --

```
# Update  
x = rev[-1]
```

```
# Update with most recent revenue value
```

```
x = rev[-1]
```

...

This line of code takes the most recent revenue value, stored in a list named rev, and assigns it to a variable named x. The code uses the -1 index feature of lists in Python to extract the last element in the list.

...

```
x = rev[-1]
```

# Principles of good documentation writing

- Be clear and concise
- Avoid redundancy
- ~~Keep comments up to date:~~

```
def calculate_total_price(item_prices, tax_rate):
    """
    Calculate the total price of items including tax.

    Parameters:
    item_prices (list of float): List of item prices.
    tax_rate (float): Tax rate to apply.

    Returns:
    float: Total price including tax.
    """
    subtotal = sum(item_prices)
    total_tax = subtotal * tax_rate
    total_price = subtotal + total_tax

    return total_price
```

# # SDK Documentation

## ## Introduction

Welcome to the SDK documentation. This guide provides an overview of how to use the SDK to interact [with](#) our API.

## ## Authentication

Before making any API requests, you need to authenticate. See the [Authentication Guide](#authentication-guide) for detailed instructions.

## ## Making Requests

To make a request, use the `ApiClient` class. Below [is](#) an example of how to fetch user data:

### ### Example: Fetch User Data

```
```python
from sdk import ApiClient
```

```
client = ApiClient(api_key='your_api_key')
```

```
# Fetch user data
```

# Principles of good documentation writing

- Be clear and concise
- Avoid redundancy
- Think of your audience
- ~~Keep comments up to date:~~

# Principles of good documentation writing

- Be clear and concise
- Avoid redundancy
- Think of your audience
- Follow language-specific conventions
- ~~Keep comments up to date:~~

```
def calculate_total_price(item_prices, tax_rate):
    """
    Calculate the total price of items including tax.

    Parameters:
    item_prices (list of float): List of item prices.
    tax_rate (float): Tax rate to apply.

    Returns:
    float: Total price including tax.
    """
    subtotal = sum(item_prices)
    total_tax = subtotal * tax_rate
    total_price = subtotal + total_tax

    return total_price
```

<https://peps.python.org/pep-0257/>

Show a quick screencast here?

## Contents

- Abstract
- Rationale
- Specification
  - What is a Docstring?
  - One-line Docstrings
  - Multi-line Docstrings
  - Handling Docstring Indentation
- Copyright
- Acknowledgements

[Page Source \(GitHub\)](#)

# PEP 257 – Docstring Conventions

**Author:** David Goodger <[goodger@python.org](mailto:goodger@python.org)>, Guido van Rossum <[guido@python.org](mailto:guido@python.org)>

**Discussions-To:** [Doc-SIG list](#)

**Status:** Active

**Type:** Informational

**Created:** 29-May-2001

**Post-History:** 13-Jun-2001

## ► Table of Contents

### Abstract

This PEP documents the semantics and conventions associated with Python docstrings.

### Rationale

The aim of this PEP is to standardize the high-level structure of docstrings: what they should contain, and how to say it (without touching on any markup syntax within docstrings). The PEP contains conventions, not laws or syntax.

*"A universal convention supplies all of maintainability, clarity, consistency, and a foundation for good programming habits too. What it doesn't do is insist that you follow it against your will. That's Python!"*

*—Tim Peters on comp.lang.python, 2001-06-16*

If you violate these conventions, the worst you'll get is some dirty looks. But some software (such as the `Docutils` docstring processing system [PEP 256](#), [PEP 258](#)) will be aware of the conventions, so following them will get you the best results.

### Specification

#### What is a Docstring?

A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. Such a docstring becomes the `__doc__` special attribute of that object.

All modules should normally have docstrings, and all functions and classes exported by a module should also have docstrings. Public methods (including the `__init__` constructor) should also have docstrings. A

# Principles of good documentation writing

- Be clear and concise
- Avoid redundancy
- Think of your audience
- Follow language-specific conventions
- Keep documentation up to date!



DeepLearning.AI

# Documentation

---

## Inline comments

```
# Initialize x with the value 10
```

```
x = 10
```

```
import pandas as pd
import numpy as np

# load weather data
weather_df = pd.DataFrame('april2024_station_data.csv')

# Numpy is faster so convert
wind_speed = df['wind_speed'].to_numpy()
wind_direction = df['wind_direction'].to_numpy()

# Better built in function in np
wind_direction_rad = np.deg2rad(wind_direction)
```

# Principles of good documentation writing

- Be clear and concise
- Avoid redundancy
- Think of your audience
- Follow language-specific conventions
- Keep documentation up to date!

```
def count_unique_letters(input_string):
    """
    Count the number of unique letters in the input string.

    Parameters:
    input_string (str): The string to analyze.

    Returns:
    int: The count of unique letters.
    """
    # Use a set to store unique letters
    unique_letters = set()

    for char in input_string:
        if char.isalpha(): # Is character a letter?
            unique_letters.add(char.lower())

    return len(unique_letters)
```

User: Experienced Developer

```
def count_unique_letters(input_string):
    """
    Count the number of unique letters in the input string.

    This function ignores case and considers only alphabetic characters.
    For example, 'A' and 'a' are treated as the same letter.

    Parameters:
    input_string (str): The string to analyze.

    Returns:
    int: The count of unique letters.
    """
    # Initialize an empty set to store unique letters
    unique_letters = set()

    # Iterate over each character in the input string
    for char in input_string:
        # Check if the character is a letter
        if char.isalpha():
            # Convert the letter to lowercase and add it to the set
            unique_letters.add(char.lower())

    # The number of unique letters is the size of the set
    return len(unique_letters)
```

Longer docstring

More inline comments

User: Novice Developer

```
def calculate_area(radius):  
    pi = 3.14159  
    return pi * radius * radius
```

```
def calculate_area(radius):  
    pi = 3.14159 # Define the value of pi  
    return pi * radius * radius # Calculate and return the area of the circle
```

```
function calculate_area(radius)
    return π * radius^2
end
```

Julia code

```
# Define a function named calculate_area that takes one parameter, radius
function calculate_area(radius)
    # Calculate the area using the formula  $\pi * \text{radius}^2$ 
    #  $\pi$  is the built-in constant for pi in Julia
    #  $\wedge$  is the exponentiation operator, so  $\text{radius}^\wedge 2$  means radius squared
    return  $\pi * \text{radius}^\wedge 2$ 
end
```

```
# Example usage:
# Call the calculate_area function with a radius of 5
area = calculate_area(5)
```

```
# Print the result
println("The area is: $area")
```

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

```
def bubble_sort(arr):
    n = len(arr) # Get the length of the array
    for i in range(n): # Traverse through all array elements
        for j in range(0, n-i-1): # Last i elements are already sorted
            if arr[j] > arr[j+1]: # Swap if the element found is greater than the next element
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr # Return the sorted array
```

```
def bubble_sort(arr):
    n = len(arr) # Get the length of the array
    for i in range(n): # Traverse through all array elements
        for j in range(0, n-i-1): # Last i elements are already sorted
            if arr[j] > arr[j+1]: # Swap if the element found is greater than the next element
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr # Return the sorted array
```

```
import pandas as pd
import numpy as np

# load weather data
weather_df = pd.DataFrame('april2024_station_data.csv')

# Numpy is faster so convert
wind_speed = df['wind_speed'].to_numpy()
wind_direction = df['wind_direction'].to_numpy()

# Better built in function in np
wind_direction_rad = np.deg2rad(wind_direction)
```



DeepLearning.AI

# Documentation

---

Documentation  
comments

# Documentation comments

- Special comments to explain purpose and usage of code
- More detailed than inline comments
- Used to generate automated documentation

# Javascript

```
/**  
 * Calculates the area of a circle given its radius.  
 *  
 * @param {number} radius - The radius of the circle.  
 * @returns {number} The area of the circle.  
 * @throws {Error} If the radius is negative.  
 *  
 * @example  
 * // Calculate the area of a circle with radius 5  
 * console.log(calculateArea(5));  
 * // Output: 78.53981633974483  
 */  
  
function calculateArea(radius) {  
    if (radius < 0) {  
        throw new Error('Radius cannot be negative');  
    }  
    return Math.PI * Math.pow(radius, 2);  
}
```

# Ruby

```
# frozen_string_literal: true

# Calculates the area of a circle given its radius.
#
# @param radius [Float] The radius of the circle.
# @return [Float] The area of the circle.
# @raise [ArgumentError] If the radius is negative.
#
# @example Calculate the area of a circle with radius 5
#   calculate_area(5)
#   # => 78.53981633974483

def calculate_area(radius)
  raise ArgumentError, 'Radius cannot be negative' if radius.negative?

  Math::PI * radius**2
end
```

```
def add(a, b):  
    """  
    Add two numbers and return the result.  
  
    Parameters:  
    a (int, float): The first number.  
    b (int, float): The second number.  
  
    Returns:  
    int, float: The sum of the two numbers.  
    """  
  
    return a + b
```

```
def add(a, b):  
    """  
    Add two numbers and return the result.  
  
    Parameters:  
    a (int, float): The first number.  
    b (int, float): The second number.  
  
    Returns:  
    int, float: The sum of the two numbers.  
    """  
  
    return a + b
```

```
def add(a, b):  
    """
```

Google style

Add two numbers and return the result.

Parameters:

a (int, float): The first number.

b (int, float): The second number.

Returns:

int, float: The sum of the two numbers.

"""

```
return a + b
```

```
def add(a, b):
    """
    Add two numbers and return the result.

    Parameters
    -----
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -----
    int or float
        The sum of the two numbers.
    """
    return a + b
```

NumPy/SciPy style

```
def add(a, b):
    """
    Add two numbers and return the result.

    :param a: The first number.
    :type a: int or float
    :param b: The second number.
    :type b: int or float
    :return: The sum of the two numbers.
    :rtype: int or float
    """

    return a + b
```

## reStructuredText (ReST) style

# Principles of good documentation writing

- Be clear and concise
- Avoid redundancy
- Think of your audience
- Follow language-specific conventions
- Keep documentation up to date!

```
def calculate_area(radius):  
    pi = 3.14159  
    return pi * radius * radius
```

```
def calculate_area(radius):
```

```
    """
```

Calculate the area of a circle given its radius.

Parameters:

radius (float): The radius of the circle.

Returns:

float: The area of the circle.

```
    """
```

```
pi = 3.14159
```

```
return pi * radius * radius
```

```
def find_max(numbers):
    max_number = numbers[0]
    for number in numbers:
        if number > max_number:
            max_number = number
    return max_number
```

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

```
def bubble_sort(arr):
```

```
    """
```

Perform bubble sort on a list of numbers.

Parameters:

arr (list of int or float): The list of numbers to sort.

Returns:

list of int or float: The sorted list of numbers.

```
    """
```

```
n = len(arr)
```

```
for i in range(n):
```

```
    for j in range(0, n-i-1):
```

```
        if arr[j] > arr[j+1]:
```

```
            arr[j], arr[j+1] = arr[j+1], arr[j]
```

```
return arr
```

```
def bubble_sort(arr):
```

```
    """
```

```
        Perform bubble sort on a list of numbers.
```

Parameters:

arr (list of int or float): The list of numbers to sort.

Returns:

list of int or float: The sorted list of numbers.

```
    """
```

```
n = len(arr)
```

```
for i in range(n):
```

```
    for j in range(0, n-i-1):
```

```
        if arr[j] > arr[j+1]:
```

```
            arr[j], arr[j+1] = arr[j+1], arr[j]
```

```
return arr
```



DeepLearning.AI

# Documentation

---

Automated  
documentation tools

## reStructuredText Style

```
def add(a, b):
    """
    Add two numbers and return the result.

    :param a: The first number.
    :type a: int or float
    :param b: The second number.
    :type b: int or float
    :return: The sum of the two numbers.
    :rtype: int or float
    """

    return a + b
```

# add module

`add.add(a, b)`

Add two numbers and return the result.

- Parameters::**
- **a** (*int or float*) – The first number.
  - **b** (*int or float*) – The second number.

**Returns::** The sum of the two numbers.

**Return type::** int or float

pip install sphinx

# sphinx-quickstart

```
def calculate_area(radius):  
    pi = 3.14159  
    return pi * radius * radius
```

```
def calculate_area(radius):
```

```
    """
```

Calculate the area of a circle given its radius.

:param radius: The radius of the circle.

:type radius: float

:return: The area of the circle.

:rtype: float

```
    """
```

```
pi = 3.14159
```

```
return pi * radius * radius
```

# Setting up Sphinx with an LLM

**Prompts:** As an expert in generating automatic documentation, can you help me set up Sphinx to generate documentation for the following python code?

```
def calculate_area(radius):
    """
    Calculate the area of a circle given its radius.

    :param radius: The radius of the circle.
    :type radius: float
    :return: The area of the circle.
    :rtype: float
    """
    pi = 3.14159
    return pi * radius * radius
```



conf.py

```
import os
import sys
sys.path.insert(0, os.path.abspath('/content/sphinx-test/source'))
```



conf.py

```
import os
import sys
sys.path.insert(0, os.path.abspath('/content/sphinx-test/source'))
extensions = ['sphinx.ext.autodoc']
```



## area module

---

.. automodule:: area

:members:



.. toctree::

:maxdepth: 2

:caption: Contents:

area

# area module

`area.calculate_area(radius)`

Calculate the area of a circle given its radius.

**Parameters::** `radius (float)` – The radius of the circle.

**Returns::** The area of the circle.

**Return type::** float



DeepLearning.AI

# Documentation

---

Autodocs with Sphinx  
walkthrough (optional)



DeepLearning.AI

# Documentation

---

Documentation in other  
coding languages

# Principles of good documentation writing

- Be clear and concise
- Avoid redundancy
- Think of your audience
- Follow language-specific conventions
- Keep documentation up to date!

# Good Java or C#

# Python

```
def calculate_total_price(item_prices, tax_rate):  
    """
```

Calculate the total price of items including tax.

Parameters:

item\_prices (list of float): List of item prices.

tax\_rate (float): Tax rate to apply.

Returns:

float: Total price including tax.

```
"""
```

```
subtotal = sum(item_prices)
```

```
total_tax = subtotal * tax_rate
```

```
total_price = subtotal + total_tax
```

```
return total_price
```

# Java

```
public class PriceCalculator {  
    /**  
     * Calculates the total price of items including tax.  
     *  
     * @param itemPrices List of item prices (in floats).  
     * @param taxRate Tax rate to apply (as a float).  
     * @return Total price including tax (as a float).  
    */  
    public static float calculateTotalPrice(List<Float> itemPrices, float taxRate) {  
        // Calculate the subtotal by summing up all item prices  
        float subtotal = 0.0f;  
        for (float price : itemPrices) {  
            subtotal += price;  
        }  
        // Calculate the total tax amount  
        float totalTax = subtotal * taxRate;  
  
        // Calculate the total price including tax  
        float totalPrice = subtotal + totalTax;  
  
        // Return the total price  
        return totalPrice;  
    }  
}
```

```
public class BubbleSort {  
    public static void bubbleSort(int[] arr) {  
        int n = arr.length;  
        for (int i = 0; i < n; i++) {  
            for (int j = 0; j < n - i - 1; j++) {  
                if (arr[j] > arr[j + 1]) {  
                    // swap arr[j] and arr[j+1]  
                    int temp = arr[j];  
                    arr[j] = arr[j + 1];  
                    arr[j + 1] = temp;  
                }  
            }  
        }  
    }  
  
    public static void main(String[] args) {  
        int[] arr = {64, 34, 25, 12, 22, 11, 90};  
        bubbleSort(arr);  
        System.out.println("Sorted array is:");  
        for (int i = 0; i < arr.length; i++) {  
            System.out.print(arr[i] + " ");  
        }  
    }  
}
```

# Creating Javadoc strings

**Prompt** Provide Javadoc comments for the following code that are clear, extensive and relevant:

```
public class BubbleSort {  
    public static void bubbleSort(int[] arr) {  
        int n = arr.length;  
        for (int i = 0; i < n; i++) {  
            for (int j = 0; j < n - i - 1; j++) {  
                if (arr[j] > arr[j + 1]) {  
                    // swap arr[j] and arr[j+1]  
                    int temp = arr[j];  
                    arr[j] = arr[j + 1];  
                    arr[j + 1] = temp;  
                }  
            }  
        }  
  
        public static void main(String[] args) {  
            int[] arr = {64, 34, 25, 12, 22, 11, 90};  
            bubbleSort(arr);  
            System.out.println("Sorted array is:");  
            for (int i = 0; i < arr.length; i++) {  
                System.out.print(arr[i] + " ");  
            }  
        }  
    }  
}
```

```
/**  
 * This class provides an implementation of the Bubble Sort algorithm.  
 * Bubble Sort is a simple sorting algorithm that repeatedly steps through the list,  
 * compares adjacent elements and swaps them if they are in the wrong order.  
 */  
public class BubbleSort {  
    /**  
     * Sorts the given array using the Bubble Sort algorithm.  
     *  
     * @param arr the array to be sorted  
     */  
    public static void bubbleSort(int[] arr) {  
        int n = arr.length;  
        // Traverse through all array elements  
        for (int i = 0; i < n; i++) {  
            // Last i elements are already sorted, no need to check them  
            for (int j = 0; j < n - i - 1; j++) {  
                // Swap if the element found is greater than the next element  
                if (arr[j] > arr[j + 1]) { ...  
    }  
}
```

```
/**  
 * This class provides an implementation of the Bubble Sort algorithm.  
 * Bubble Sort is a simple sorting algorithm that repeatedly steps through the list,  
 * compares adjacent elements and swaps them if they are in the wrong order.  
 */  
  
public class BubbleSort {  
    /**  
     * Sorts the given array using the Bubble Sort algorithm.  
     *  
     * @param arr the array to be sorted  
     */  
  
    public static void bubbleSort(int[] arr) {  
        int n = arr.length;  
        // Traverse through all array elements  
        for (int i = 0; i < n; i++) {  
            // Last i elements are already sorted, no need to check them  
            for (int j = 0; j < n - i - 1; j++) {  
                // Swap if the element found is greater than the next element  
                if (arr[j] > arr[j + 1]) { ...  
    }  
}
```

```
/**  
 * The main method is the entry point of the program.  
 * It initializes an array, sorts it using Bubble Sort, and prints the sorted array.  
 *  
 * @param args the command line arguments  
 */  
  
public static void main(String[] args) {  
    int[] arr = {64, 34, 25, 12, 22, 11, 90};  
    // Print the unsorted array  
    System.out.println("Unsorted array:");  
    for (int i = 0; i < arr.length; i++) {  
        System.out.print(arr[i] + " ");  
    }  
    System.out.println();  
    // Sort the array using Bubble Sort  
    bubbleSort(arr);
```

# BubbleSort

This class provides an implementation of the Bubble Sort algorithm. Bubble Sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order.

## Methods

- [bubbleSort](#)
- [main](#)

### **bubbleSort**

```
public static void bubbleSort(int[] arr)
```

Sorts the given array using the Bubble Sort algorithm.

#### **Parameters:**

arr - the array to be sorted

### **main**

```
public static void main(String[] args)
```

The main method is the entry point of the program. It initializes an array, sorts it using Bubble Sort, and prints the sorted array.

#### **Parameters:**

args - the command line arguments

```
function bubbleSort(arr) {  
    let n = arr.length;  
    for (let i = 0; i < n; i++) {  
        for (let j = 0; j < n - i - 1; j++) {  
            if (arr[j] > arr[j + 1]) {  
                let temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
            }  
        }  
    }  
}  
  
let arr = [64, 34, 25, 12, 22, 11, 90];  
console.log("Unsorted array:", arr);  
bubbleSort(arr);  
console.log("Sorted array is:", arr);
```

# Creating Javascript docstrings

**Prompt** Add clear and extensive comments in JSDocs format to the following code:

```
function bubbleSort(arr) {  
    let n = arr.length;  
    for (let i = 0; i < n; i++) {  
        for (let j = 0; j < n - i - 1; j++) {  
            if (arr[j] > arr[j + 1]) {  
                let temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
            }  
        }  
    }  
}  
  
let arr = [64, 34, 25, 12, 22, 11, 90];  
console.log("Unsorted array:", arr);  
bubbleSort(arr);  
console.log("Sorted array is:", arr);
```

```
/**  
 * Sorts the given array using the Bubble Sort algorithm.  
 *  
 * @param {number[]} arr - The array to be sorted  
 */  
function bubbleSort(arr) {  
    let n = arr.length;  
    // Traverse through all array elements  
    for (let i = 0; i < n; i++) {  
        // Last i elements are already sorted, no need to check them  
        for (let j = 0; j < n - i - 1; j++) {  
            // Swap if the element found is greater than the next element  
            if (arr[j] > arr[j + 1]) {  
                let temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
            }  
        }  
    }  
}
```

## Methods

### `bubbleSort(arr)`

Sorts the given array using the Bubble Sort algorithm.

Parameters:

Name	Type	Description
<code>arr</code>	<code>Array.&lt;number&gt;</code>	The array to be sorted

Source:

[jsdoc-example.js, line 6](#)

---

Documentation generated by [JSDoc 4.0.3](#) on Tue Jul 09 2024 11:37:21 GMT-0700 (Pacific Daylight Time)

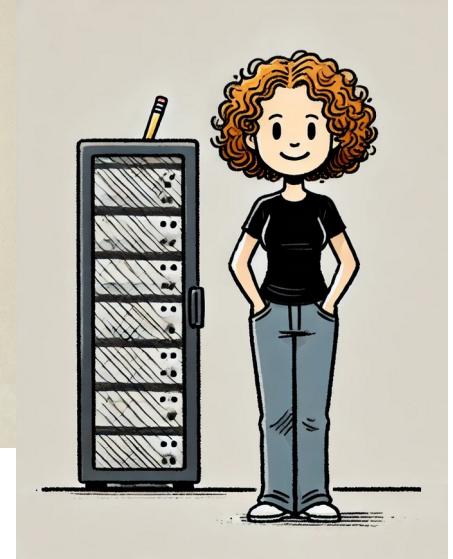
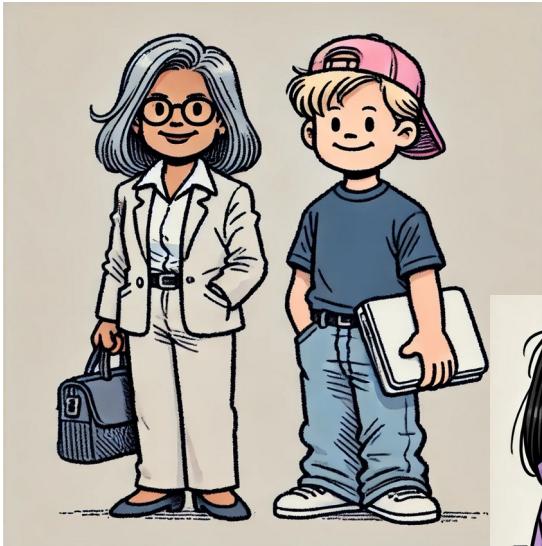


DeepLearning.AI

# Documentation

---

Documentation for life  
in production



Cloud Infrastructure  
Changes



Module Dependency  
Updates



Web specification  
Releases



Code Refactoring





DeepLearning.AI

# Dependency Management

---

## Module introduction

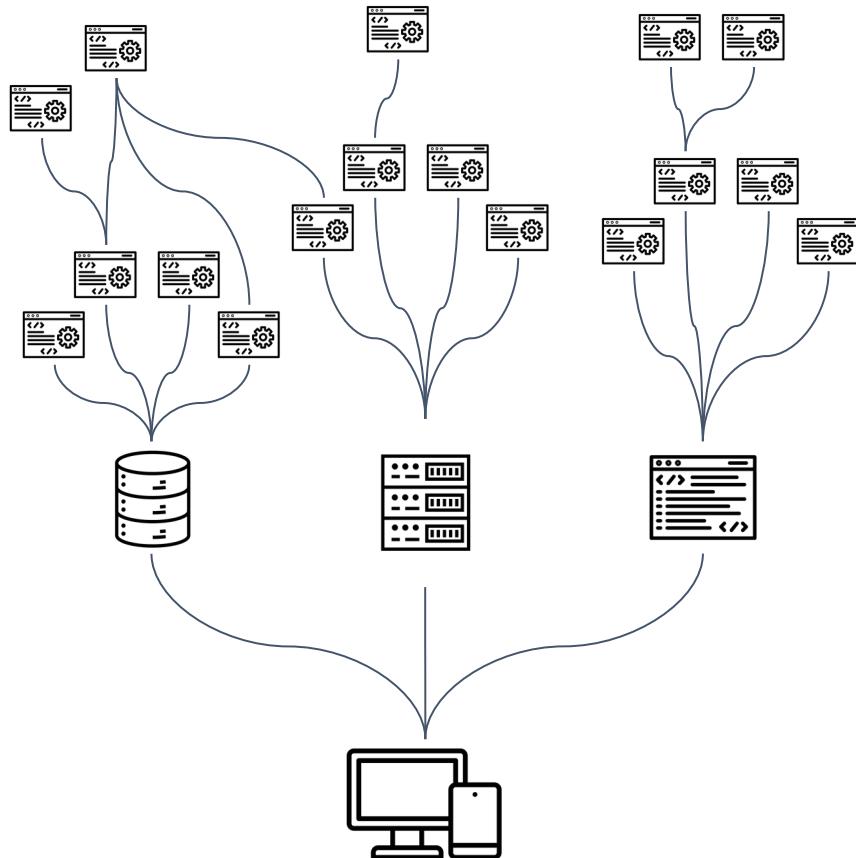
# Dependencies

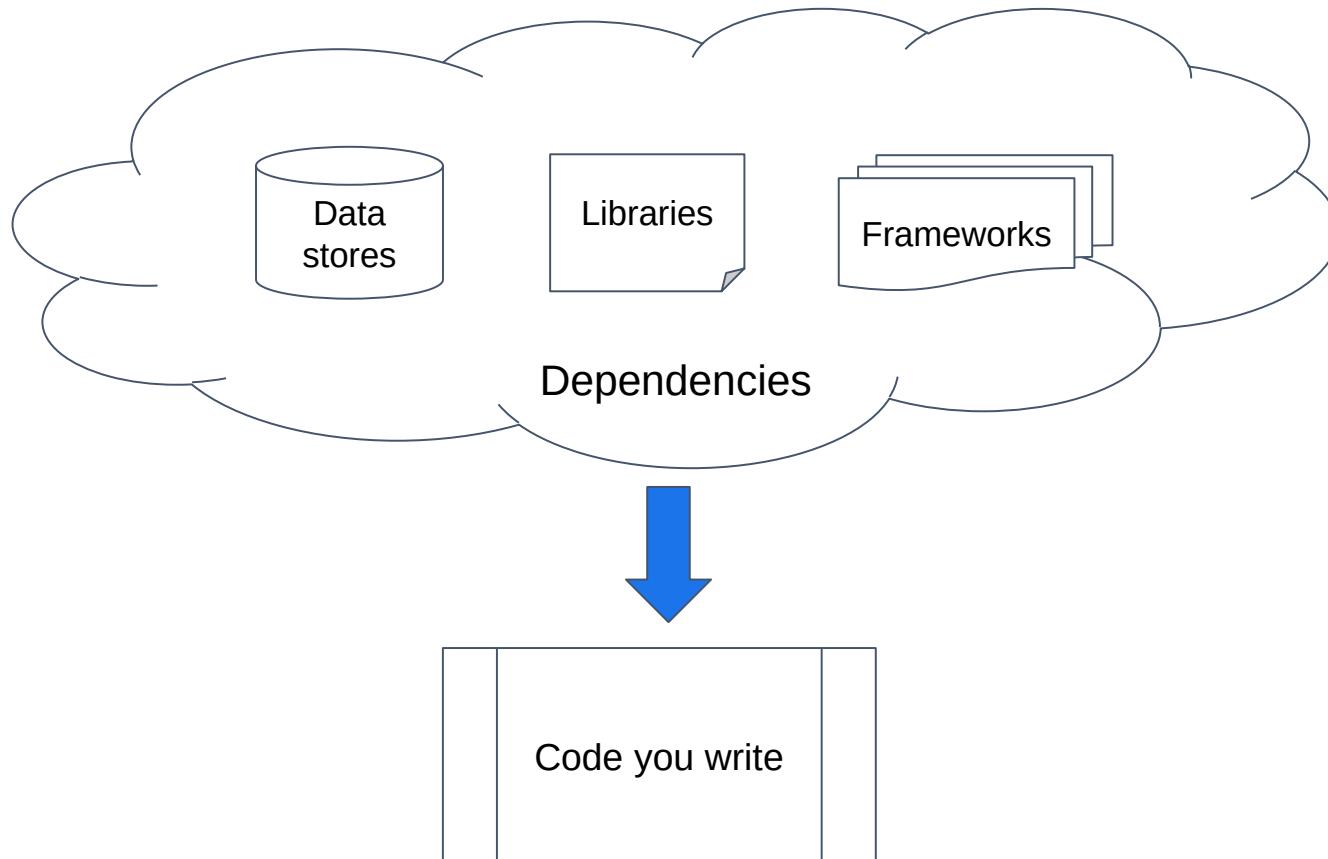
## Pros

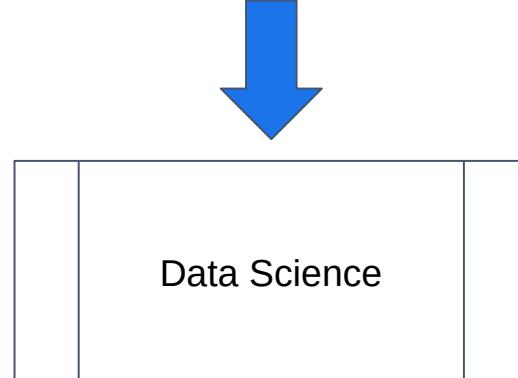
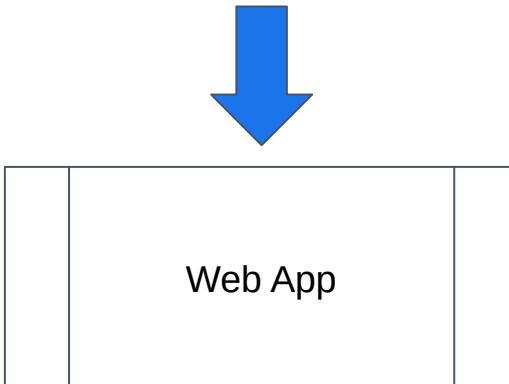
- Off-the-shelf solutions
- Often efficient, tested, secure

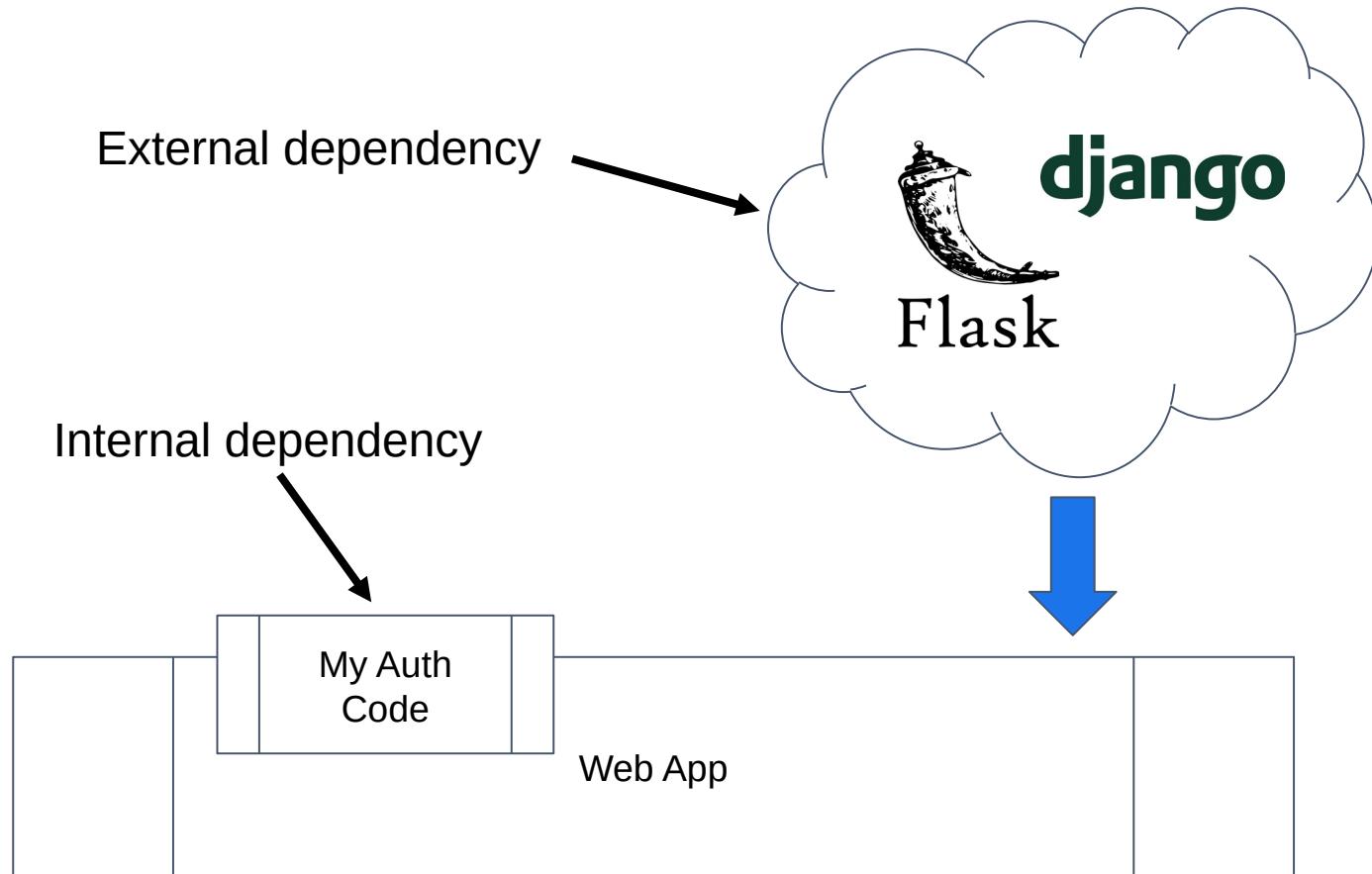
## Cons

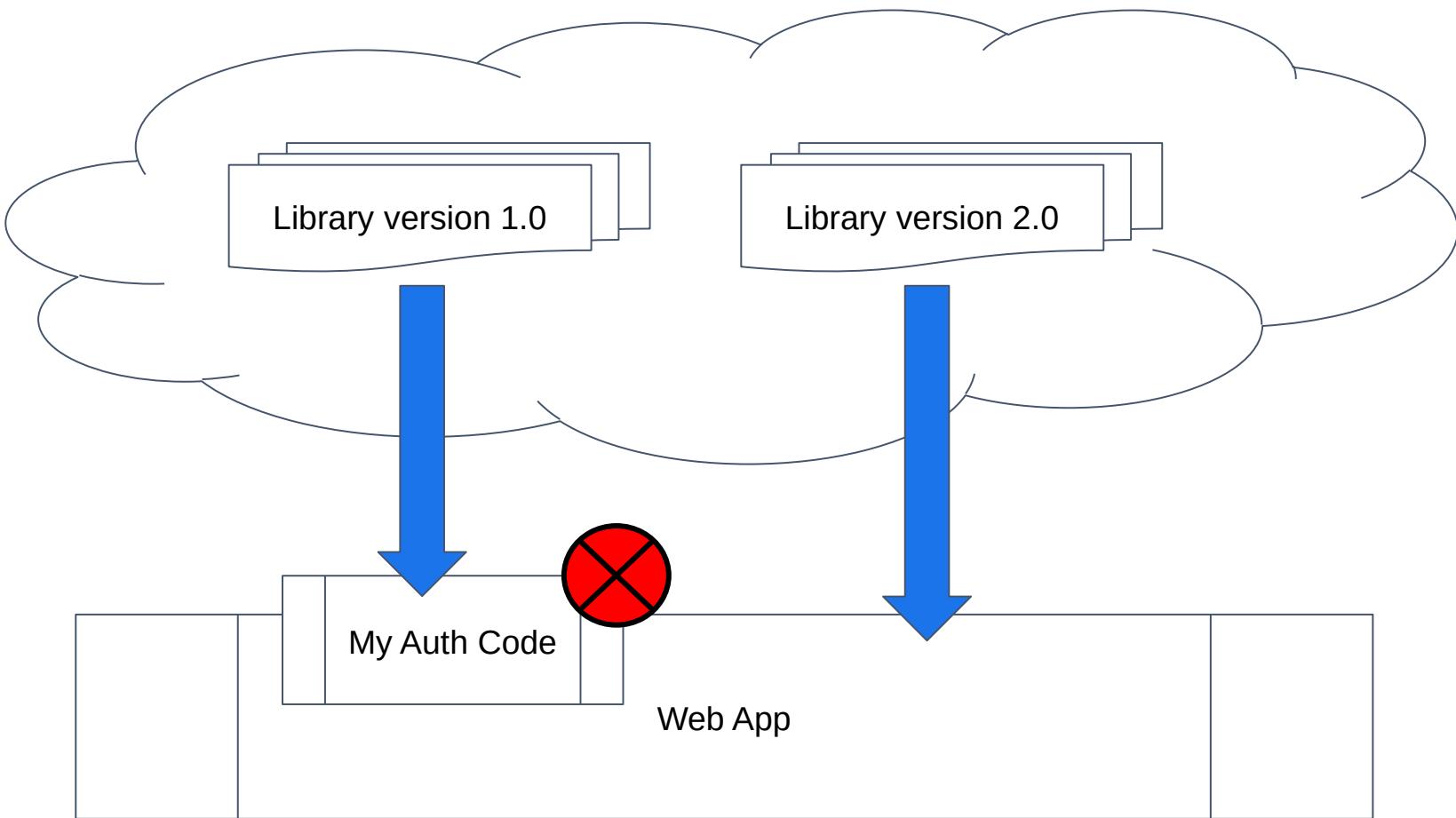
- Your code “depends” on other code to work as expected

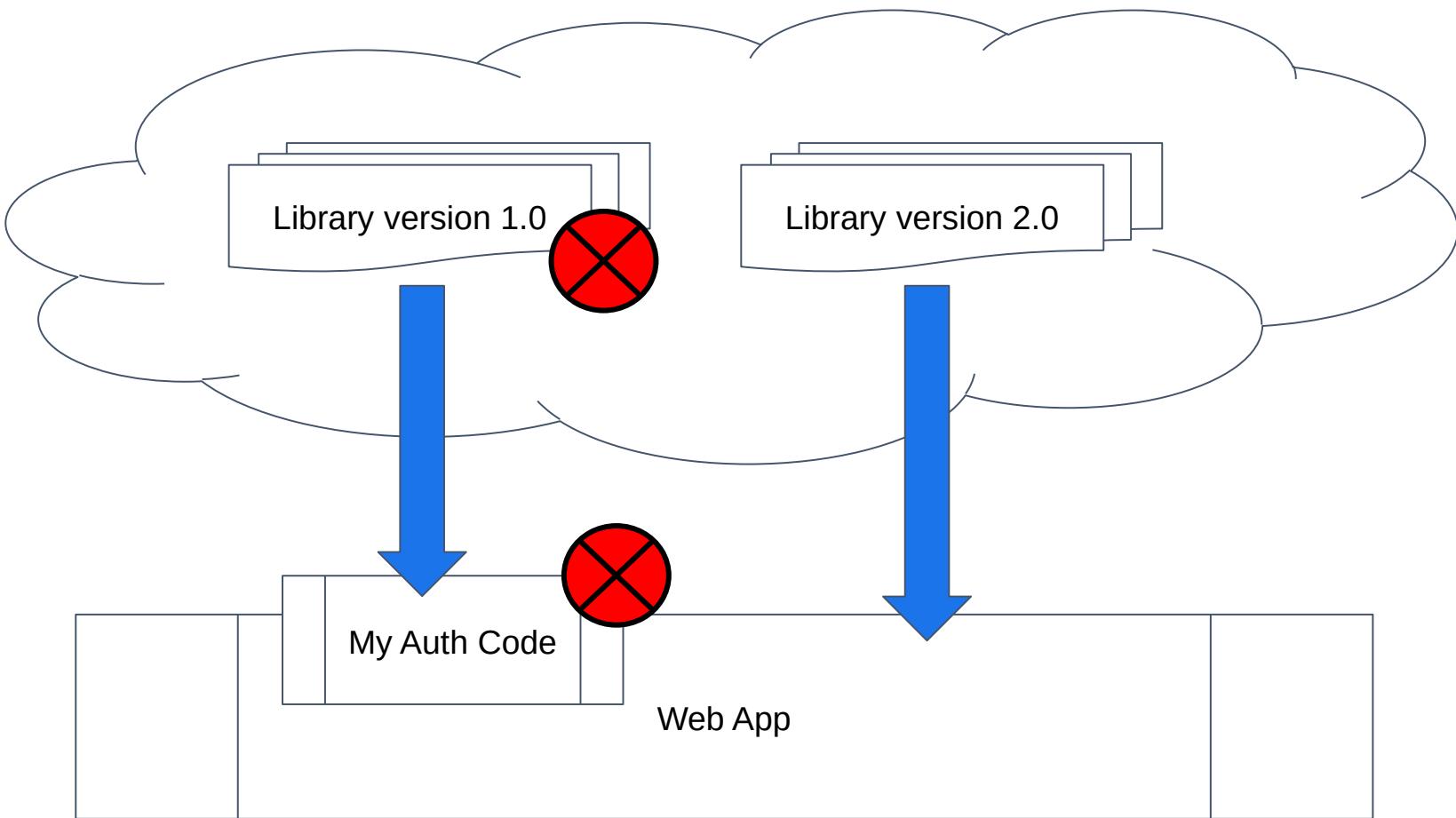


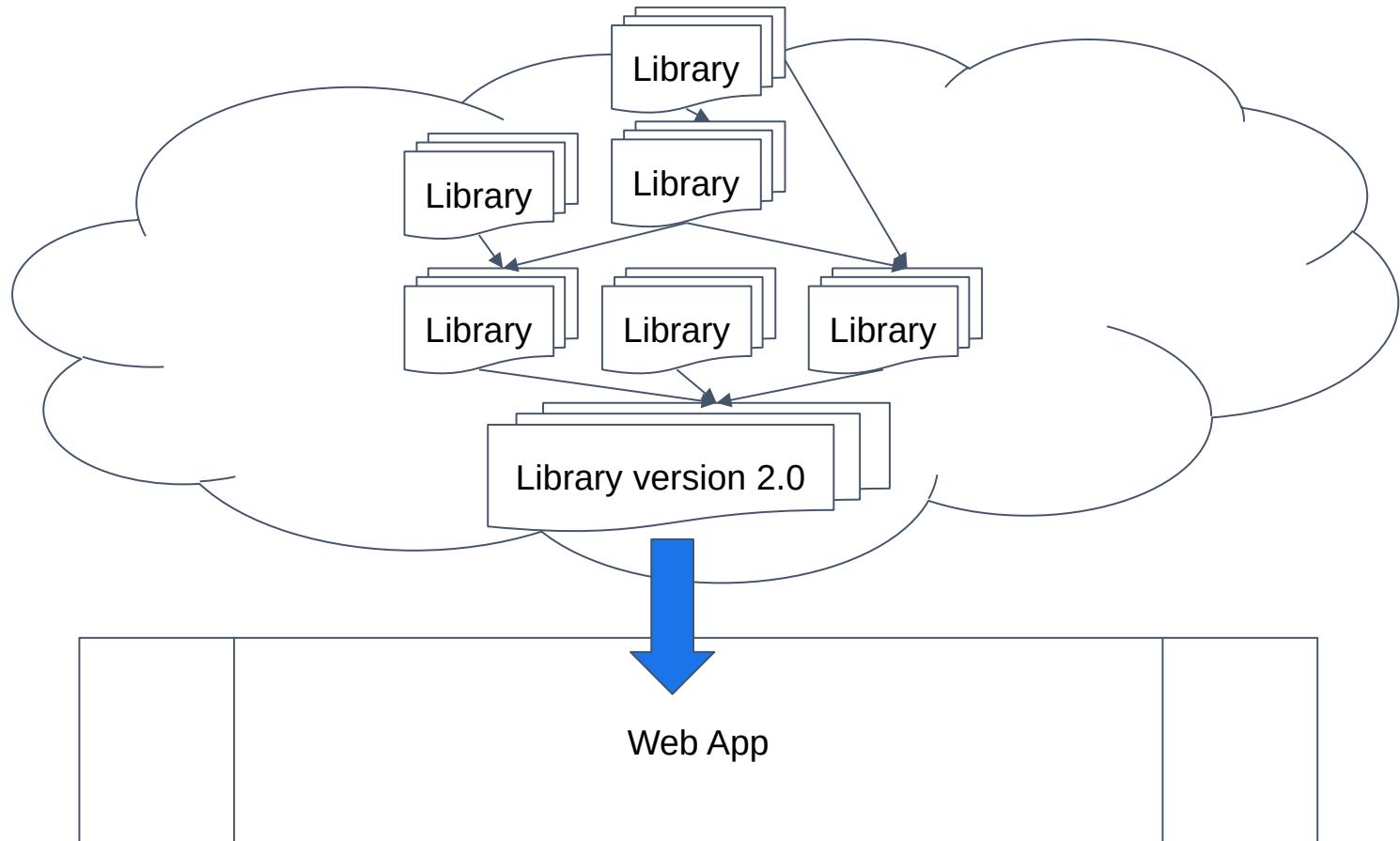












# LLMs and dependencies

## Strengths

- Brainstorm libraries and packages to use for your project
- Learn more about a dependency
- Identify dependency conflicts
- Suggest solutions to issues with dependencies

## Weaknesses

- LLMs may not know about dependency changes past their training date (some models do search the web)
- May be less helpful or accurate with obscure libraries that are rarely in their training data

# Try it yourself!

Have a discussion with an LLM about a recent software project you've worked on:

- Ask it to suggest some libraries to consider
- Follow up to learn more about one of the libraries. What does it do? What are the pros and cons of choosing it? How does it compare to alternatives?
- Don't forget your prompting skills
  - Be specific
  - Assign a role
  - Request an expert opinion
  - Give feedback



DeepLearning.AI

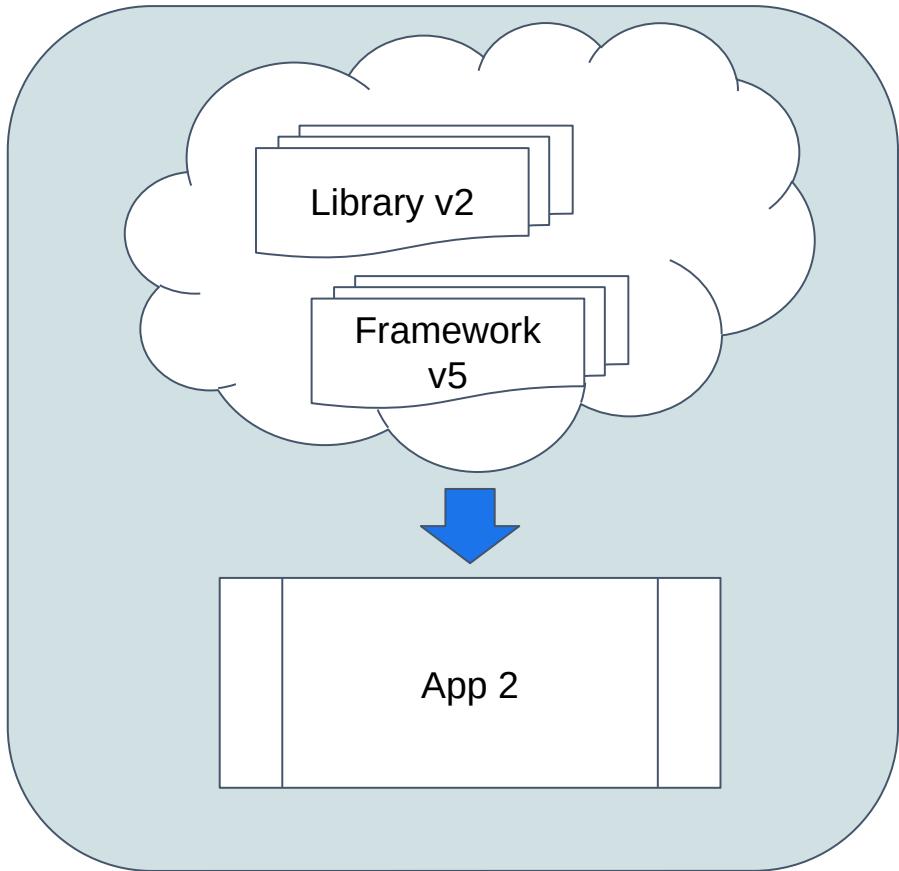
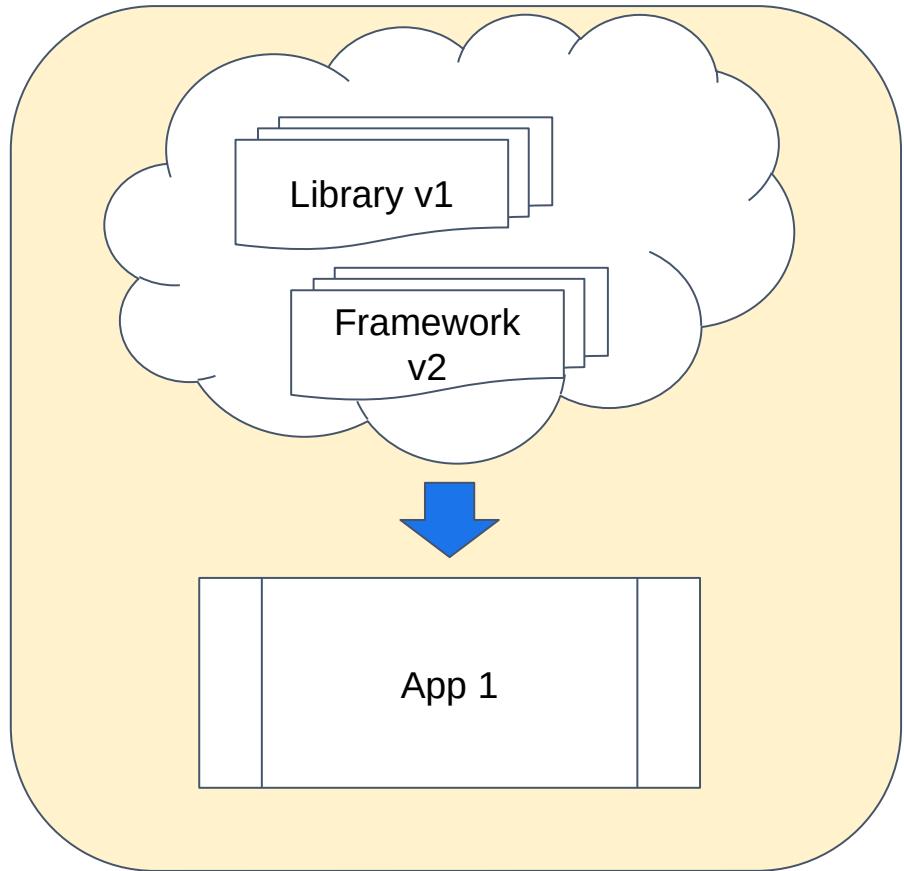
# Dependency Management

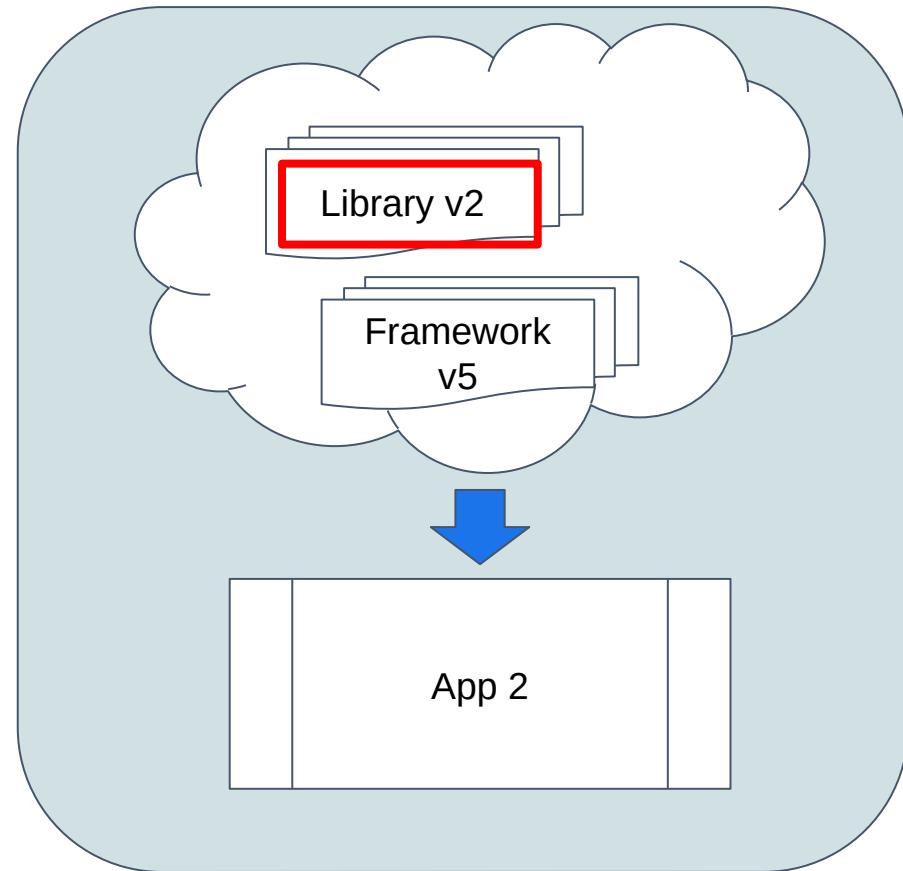
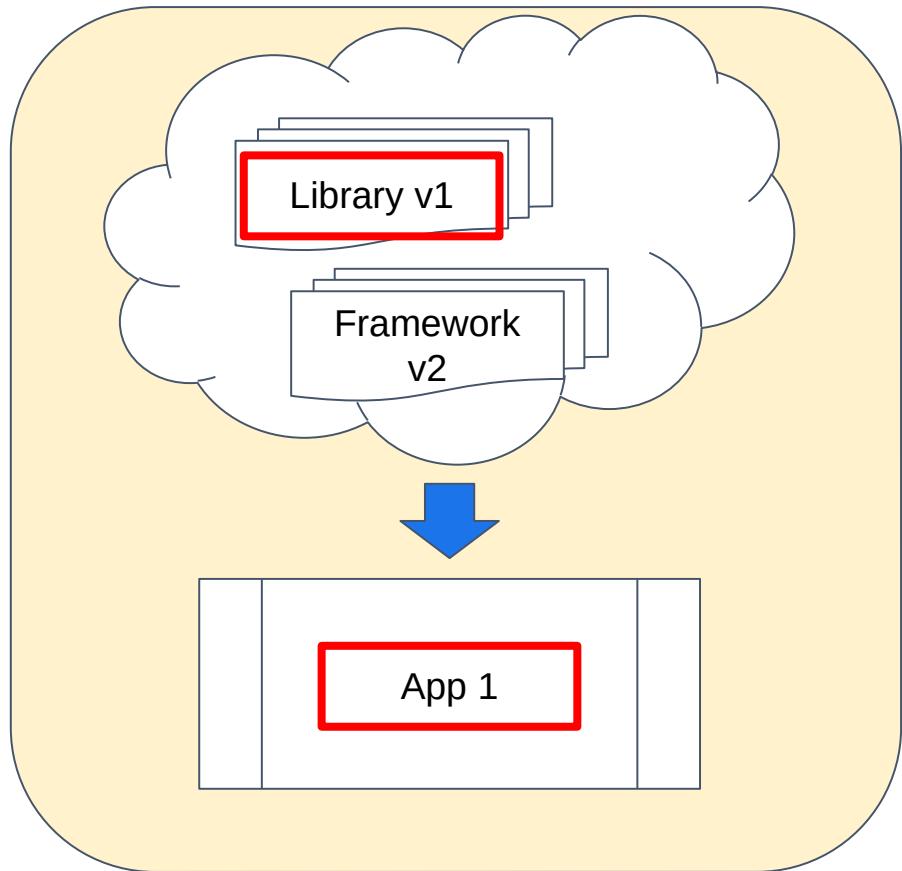
---

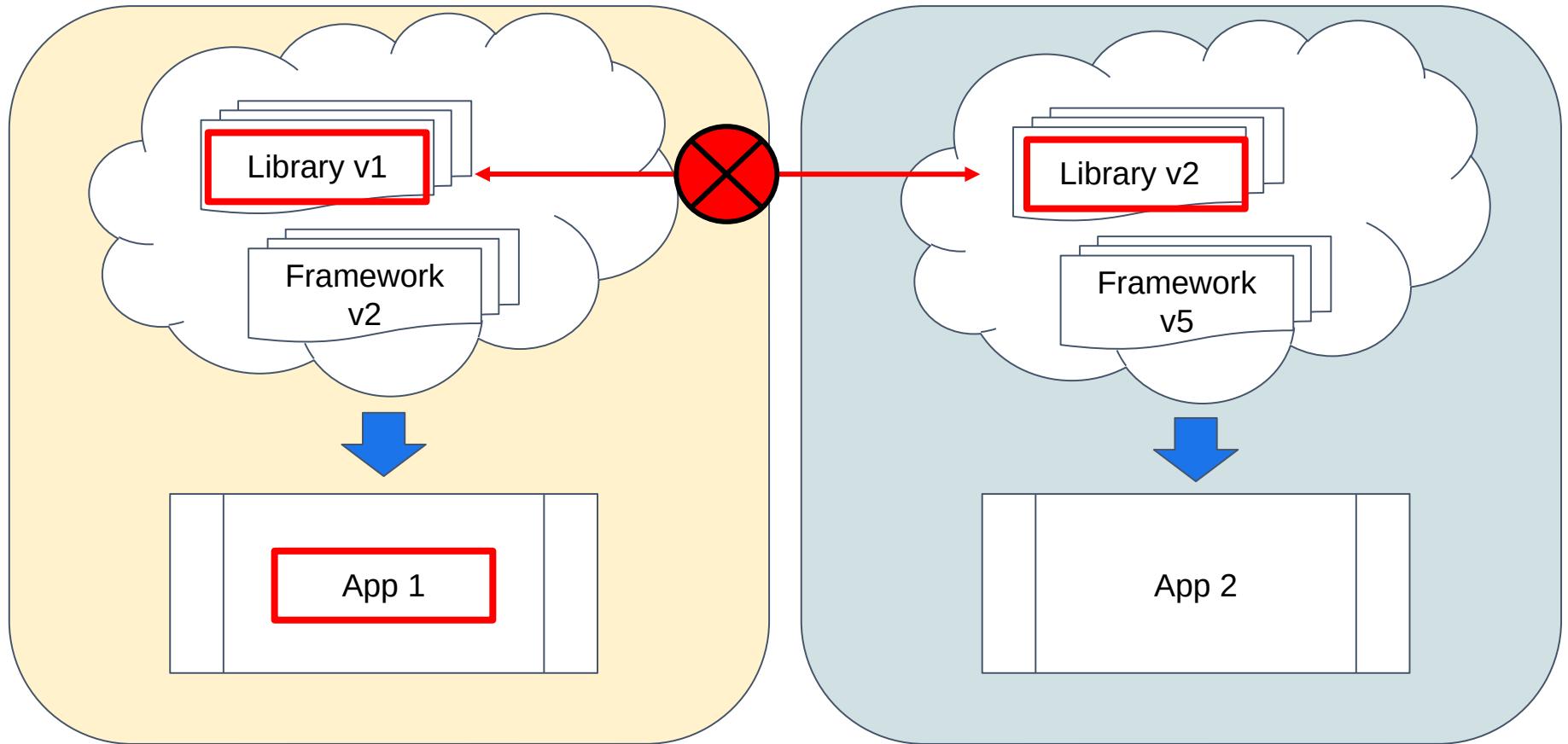
## Virtual environments

# Virtual environments

- Isolated workspaces that allow you to manage dependencies for a specific project, independently of others
- Isolation is crucial to avoid dependency conflicts between projects







# Benefits of virtual environments

- **Isolation:** Each project has its own dependencies, avoiding conflicts
- **Reproducibility:** Ensures that the project runs with the same dependencies across different environments
- **Manageability:** Easier to manage and update dependencies for individual projects

# Instructions to set up a virtual environment

**Prompt:** You're an expert in Python virtual environment setup. What steps can I follow to set up a virtual environment on my Mac using venv?

## 1. Ensure Python is Installed

- MacOS typically comes with Python pre-installed. However, you may want to use Python 3 if it's not already the default.
- Verify Python installation:

```
bash
```

 Copy code

```
python3 --version
```

- If Python 3 isn't installed, you can install it using [Homebrew](#):

```
bash
```

 Copy code

```
brew install python
```

## 2. Navigate to Your Project Directory

- Open the terminal and navigate to the directory where you want to create your virtual environment:

```
bash
```

 Copy code

```
cd /path/to/your/project
```

### 3. Create a Virtual Environment

- Use Python's built-in `venv` module to create a virtual environment:

```
bash
```

 Copy code

```
python3 -m venv myenv
```

- Here, `myenv` is the name of your virtual environment. You can replace it with any name you like.

### 4. Activate the Virtual Environment

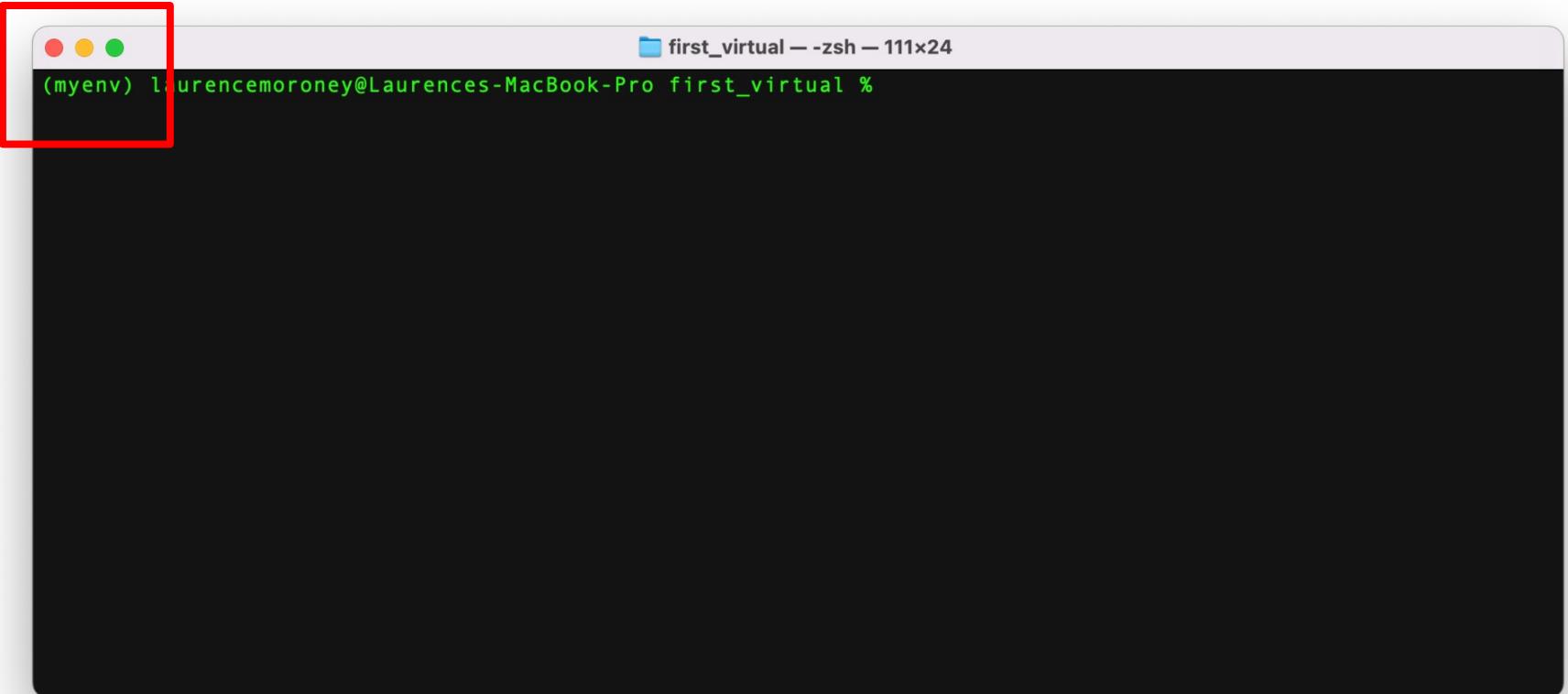
- To activate the virtual environment, use the following command:

```
bash
```

 Copy code

```
source myenv/bin/activate
```

- Once activated, you'll see the environment's name in your terminal prompt.



The image shows a screenshot of a macOS terminal window. The window title is "first\_virtual — -zsh — 111x24". The prompt "(myenv) laurencemoroney@Laurences-MacBook-Pro first\_virtual %" is displayed in green text. The top-left corner of the window frame is highlighted with a red rectangle. The main body of the terminal is a large black area.

## 5. Install Project Dependencies

- With the environment activated, you can install your project's dependencies using `pip`:

```
bash
```

 Copy code

```
pip install <package-name>
```

- You can also install dependencies from a `requirements.txt` file:

```
bash
```

 Copy code

```
pip install -r requirements.txt
```

## 6. View Installed Packages

- To see a list of installed packages within your virtual environment, use:

```
bash
```

 Copy code

```
pip list
```

## 7. Deactivate the Virtual Environment

- When you're done working in the virtual environment, you can deactivate it by running:

```
bash
```

 Copy code

```
deactivate
```

## 8. Reactivating the Virtual Environment

- Anytime you want to reactivate the environment in the future, just navigate to the project directory and run:

```
bash
```

 Copy code

```
source myenv/bin/activate
```

# Try it yourself!

- Create two virtual environments, env1 and env2
- In env1, install requests with this command  
`pip install requests`
- In env2, install requests with this command  
`pip install requests==2.31.0`
- Use the source and deactivate commands to move in and out of each environment and pip list to see the different version of requests

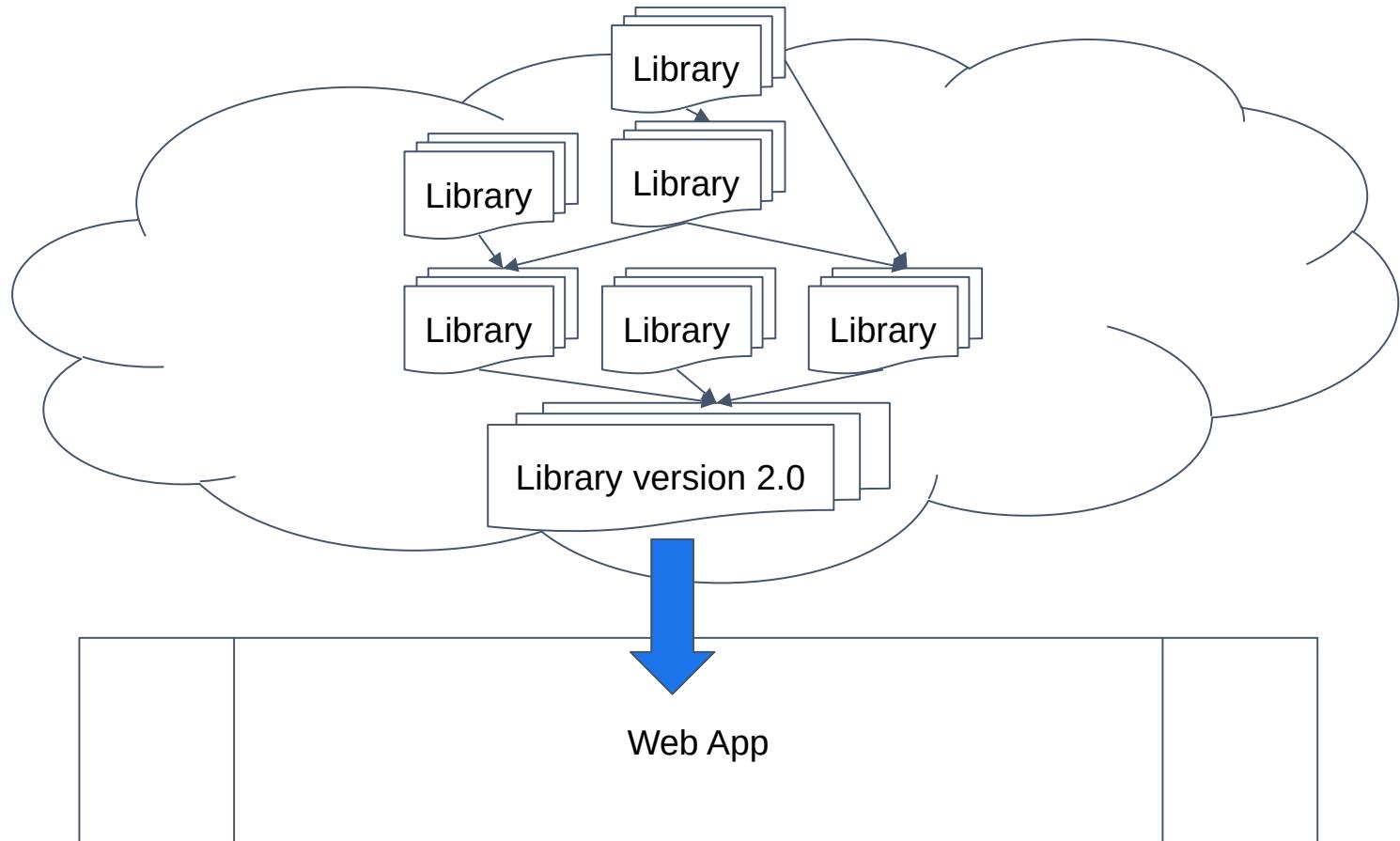


DeepLearning.AI

# Dependency Management

---

## Researching dependencies



# Managing dependencies in different languages



Python



Javascript



C#

pip list

pip freeze

certifi==2024.6.2

charset-normalizer==3.3.2

idna==3.7

requests==2.32.3

urllib3==2.2.1

“pinning”

# Common commands with pip-tools

`pip-compile` : converts a list of direct dependencies (usually named `requirements.in`) into a list of all transitive dependencies (usually named `requirements.txt`)

`pip-sync` : install or update libraries in `requirements.txt`

# Preparing dependencies using pip compile



requirements.in

pip compile



requirements.txt



- List of direct dependencies

# Installing dependencies using pip sync



requirements.txt

pip sync



```
GenAI4SD -- zsh -- 80x24
Package           Version
aiofiles          23.2.1
altair             5.3.0
annotated-types   0.6.0
anyio              4.3.0
appnope            0.1.4
argon2-cffi       23.1.0
argon2-cffi-bindings 21.2.0
arrow               1.3.0
astroid              3.1.0
asttokens            2.4.1
async-lru             2.0.4
attrs                23.2.0
Babel               2.14.0
beautifulsoup4      4.12.3
bleach               6.1.0
blinker              1.7.0
bs4                  0.0.2
cachetools            5.3.3
certifi              2024.2.2
cffi                 1.16.0
charset-normalizer    3.3.2
click                 8.1.7
```

Install packages listed in requirements.txt in env

# Instructions to set up pip-tools

**Prompt:** You're an expert in Python programming and dependency management. Please guide me on how to install pip-tools and compile my first set of dependencies.



requirements.in

requests

flask

# pip-compile requirements.in

```
[(myenv) laurencemoroney@Laurences-MacBook-Pro first_virtual % pip-compile requirements.in]
WARNING: --strip-extras is becoming the default in version 8.0.0. To silence this warning, either use --strip-extras to opt into the new default or use --no-strip-extras to retain the existing behavior.
#
# This file is autogenerated by pip-compile with Python 3.12
# by the following command:
#
#     pip-compile requirements.in
#
blinker==1.8.2
    # via flask
certifi==2024.6.2
    # via requests
charset-normalizer==3.3.2
    # via requests
click==8.1.7
    # via flask
flask==3.0.3
    # via -r requirements.in
idna==3.7
    # via requests
itsdangerous==2.2.0
    # via flask
jinja2==3.1.4
    # via flask
markupsafe==2.1.5
    # via
    # jinja2
    # werkzeug
requests==2.32.3
    # via -r requirements.in
urllib3==2.2.1
    # via requests
werkzeug==3.0.3
    # via flask
(myenv) laurencemoroney@Laurences-MacBook-Pro first_virtual %
```



## requirements.txt

```
blinker==1.8.2                                itsdangerous==2.2.0
  # via flask
certifi==2024.6.2                               jinja2==3.1.4
  # via flask
  # via requests
charset-normalizer==3.3.2                      markupsafe==2.1.5
  # via
  # via requests
click==8.1.7                                    # werkzeug
  # via flask
flask==3.0.3                                     requests==2.32.3
  # via -r requirements.in
idna==3.7                                       urllib3==2.2.1
  # via requests
  # via requests
werkzeug==3.0.3
```



## requirements.txt

```
blinker==1.8.2
# via flask
certifi==2024.6.2
# via requests
charset-normalizer==3.3.2
# via requests
click==8.1.7
# via flask
flask==3.0.3
# via -r requirements.in
idna==3.7
# via requests
itsdangerous==2.2.0
# via flask
jinja2==3.1.4
# via flask
markupsafe==2.1.5
# via
# jinja2
# werkzeug
requests==2.32.3
# via -r requirements.in
urllib3==2.2.1
# via requests
werkzeug==3.0.3
```

# Instructions to set up pip-tools

**Prompt:** I have a dependency on a Python library called “Itsdangerous”: What does this do, and should I trust it?

I have a dependency on a Python library called 'Its-dangerous' --  
what does this do, and should I trust it?



The `itsdangerous` library is a Python package that provides various utilities for cryptographically signing and serializing data. It's particularly useful for ensuring the integrity of data passed between systems or within different components of an application, making it a common choice for handling security-sensitive information, such as tokens for web applications.

## Key Features of `itsdangerous`

1. **Data Signing:** It allows you to sign data with a secret key so that you can verify its authenticity later. This is particularly useful for ensuring that the data hasn't been tampered with.
2. **Timed Signatures:** The library supports creating signatures that expire after a certain period, which is helpful for creating time-limited tokens (e.g., password reset tokens).
3. **JSON Web Tokens (JWT):** While not a full JWT implementation, `itsdangerous` can be used to create and verify simple token-like structures that are similar to JWTs.
4. **Compact and Secure:** The signatures created by `itsdangerous` are compact and designed to be secure, using modern cryptographic techniques.

GitHub Repository: <https://github.com/pallets/itsdangerous>

PyPI Page: <https://pypi.org/project/itsdangerous/>

# Summarize a Project's Requirements

**Prompt:** I am working on a simple web application in Python. Below are the dependencies currently listed for that project. For each one write me a short summary of what it does and whether I should trust it.

```
blinker==1.8.2
    # via flask
certifi==2024.6.2
    # via requests
charset-normalizer==3.3.2
    # via requests
click==8.1.7
    # via flask
flask==3.0.3
    # via -r requirements.in
idna==3.7
    # via requests
itsdangerous==2.2.0
    # via flask
...
...
```

Sure! Here is a summary of each dependency along with an evaluation of whether you should trust it:

1. **blinker==1.8.2**

- **Summary:** Blinker is a fast, simple object-to-object and broadcast signaling library. It is often used in web applications for event handling.
- **Trustworthiness:** Yes, you can trust it. Blinker is a well-known library in the Python community, particularly for use with Flask.

2. **certifi==2024.6.2**

- **Summary:** Certifi provides Mozilla's carefully curated collection of Root Certificates for validating the trustworthiness of SSL certificates while verifying the identity of TLS hosts.
- **Trustworthiness:** Yes, you can trust it. Certifi is widely used in the Python ecosystem to ensure secure HTTP requests.

3. **charset-normalizer==3.3.2**

- **Summary:** Charset-Normalizer is an alternative to Chardet, used for detecting and normalizing character encodings.
- **Trustworthiness:** Yes, you can trust it. It's commonly used in web scraping and HTTP libraries like Requests.

4. **click==8.1.7**

## pip-sync requirements.txt

- Installs, updates, and uninstall all packages in requirements.txt to make your environment match
- Careful! Easy to mistakenly remove libraries you want!

# Try it yourself!

- Create an environment called env1
- Create a requirements.in file with at least 2 Python packages (if you need ideas, ask your LLM)
- Use pip-compile to generate a requirements.txt file
- Use an LLM to learn more about all the packages listed
- Create a second environment called env2
- Using pip-sync, copy the dependencies of env1 over to env2
- If you get stuck, use your LLM for help

```
{  
  "name": "chatgpt-clone",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "dotenv": "^16.4.5",  
    "express": "^4.19.2",  
    "openai": "^4.52.3",  
    "socket.io": "^4.7.5"  
  }  
}
```

```
{  
  "name": "chatgpt-clone",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "licence": "ISC",  
  "dependencies": {  
    "dotenv": "^16.4.5",  
    "express": "^4.19.2",  
    "openai": "^4.52.3",  
    "socket.io": "^4.7.5"  
  }  
}
```

> npm audit

# > npm outdated

```
[laurence@Laurences-Mini chatgpt-clone % npm outdated
Package  Current  Wanted  Latest  Location          Depended by
openai    4.52.3   4.52.4   4.52.4  node_modules/openai  chatgpt-clone
laurence@Laurences-Mini chatgpt-clone % ]
```

# > npm outdated

```
[laurence@Laurences-Mini chatgpt-clone % npm outdated
Package  Current  Wanted  Latest  Location          Depended by
openai    4.52.3   4.52.4   4.52.4  node_modules/openai  chatgpt-clone
laurence@Laurences-Mini chatgpt-clone % ]
```

# > npm install openai@latest

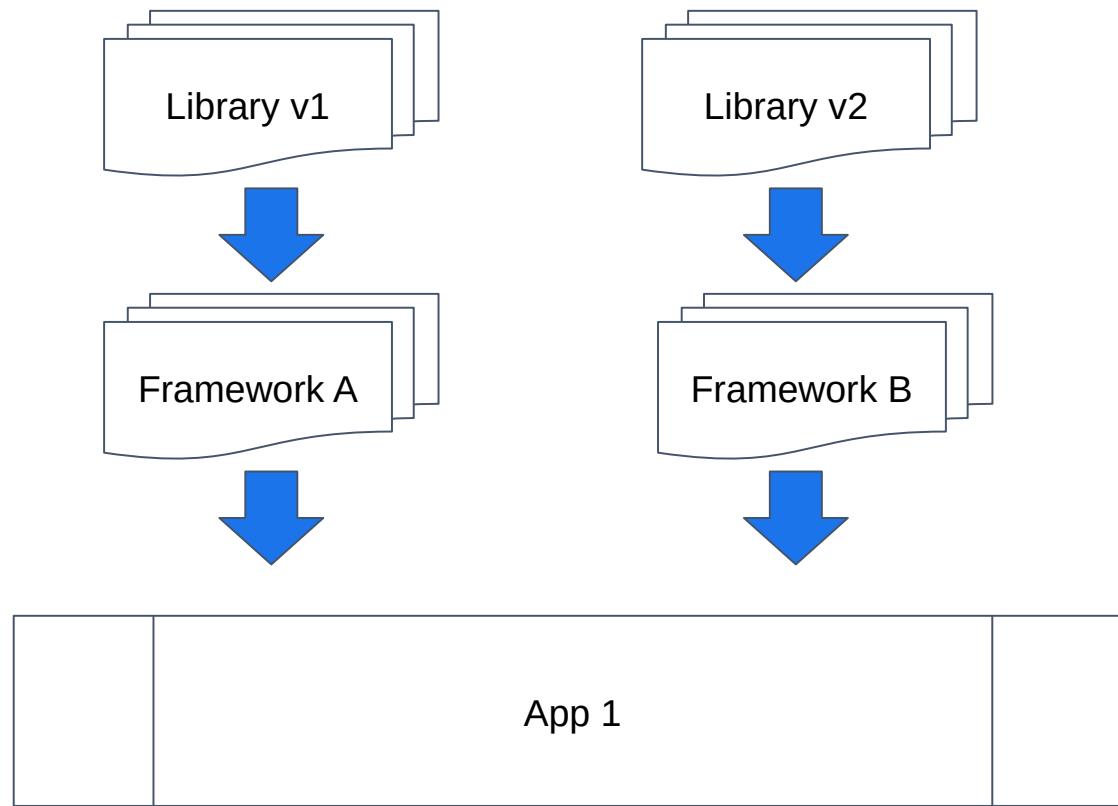


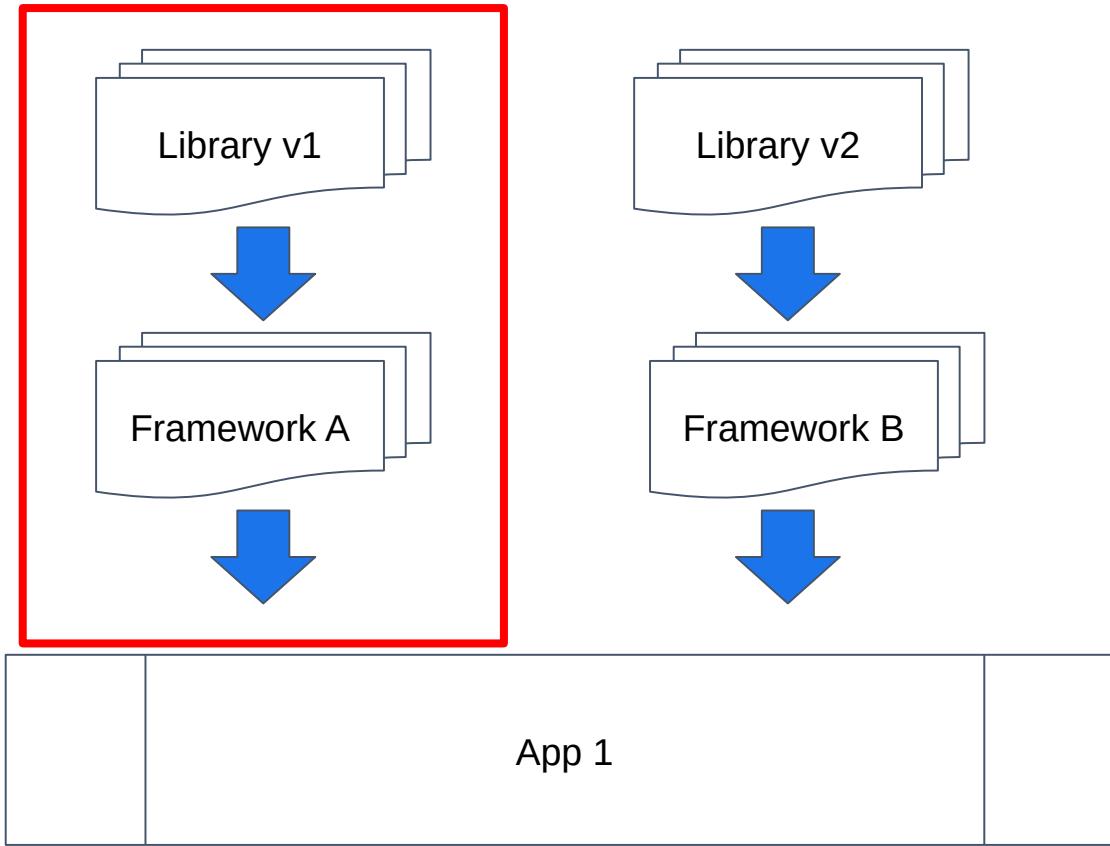
DeepLearning.AI

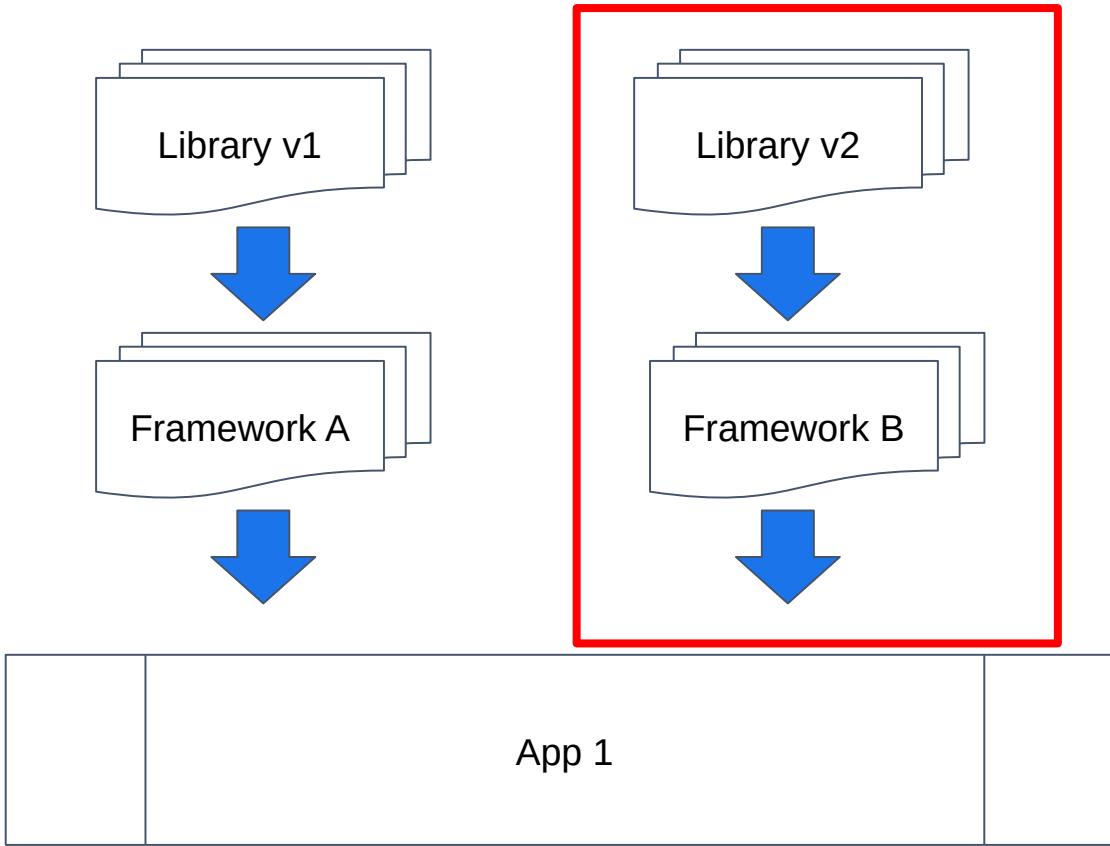
# Dependency Management

---

## Dependency conflicts







# Managing dependency conflicts

1. Identify the conflict: Determine which dependencies are causing the issue.
2. Check compatibility: Look for versions that are compatible with all packages.
3. Update dependencies: Update or modify your dependencies to resolve the conflict.

# Identifying conflicts with an LLM

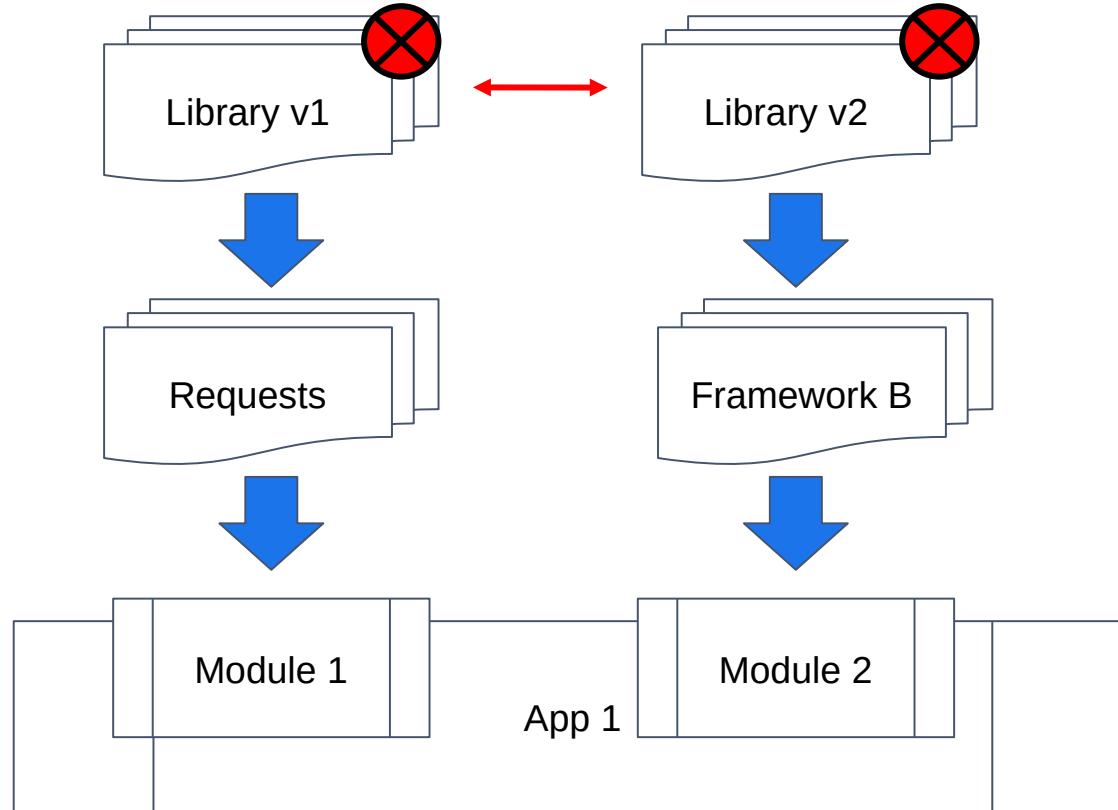
**Prompt:** I am working on a simple web application in Python. Below are the dependencies currently listed for that project. Are there any known dependency conflicts between the versions of the libraries I am using?

```
blinker==1.8.2
    # via flask
certifi==2024.6.2
    # via requests
charset-normalizer==3.3.2
    # via requests
click==8.1.7
    # via flask
flask==3.0.3
    # via -r requirements.in
idna==3.7
    # via requests
itsdangerous==2.2.0
    # via flask
...
...
```

```
import requests
```

```
def fetch_url_content(url):
    try:
        response = requests.get(url)
        response.raise_for_status() # Check if the request was successful
        print(response.text)
    except requests.exceptions.HTTPError as http_err:
        print(f"HTTP error occurred: {http_err}")
    except Exception as err:
        print(f"An error occurred: {err}")

if __name__ == "__main__":
    url = input("Enter the URL to fetch content from: ")
    fetch_url_content(url)
```



What's an alternative to the `requests` library in Python?



An alternative to the `requests` library in Python is the `httpx` library. `httpx` provides an HTTP client for Python that is similar to `requests`, but it includes additional features like asynchronous support, HTTP/2, and connection pooling.

🔉 🔍 ⌂ 🗑️ ↻



Message ChatGPT



ChatGPT can make mistakes. Check important info.



```
import httpx

def fetch_url_content(url):
    try:
        response = httpx.get(url)
        response.raise_for_status() # Check if the request was successful
        print(response.text)
    except httpx.HTTPStatusError as http_err:
        print(f"HTTP error occurred: {http_err}")
    except Exception as err:
        print(f"An error occurred: {err}")

if __name__ == "__main__":
    url = input("Enter the URL to fetch content from: ")
    fetch_url_content(url)
```

# Try it yourself! [NOT YET DEVELOPED]

- Use an LLM to help you resolve the dependency conflict in the provided environment



DeepLearning.AI

# Dependency Management

---

Dependencies and  
security

# Security issues with dependencies

- **Outdated packages:** Using old versions of packages with known vulnerabilities.
- **Transitive dependencies:** Security risks in dependencies of your dependencies.
- **Unmaintained packages:** Relying on packages that are no longer actively maintained.

```
pip install flask==1.0.0
```

# Researching package security

**Prompt:** How can I check that the packages in my current Python environment are secure?



```
(myenv) laurencemoroney@Laurences-MacBook-Pro first_virtual % pip-audit
Found 1 known vulnerability in 1 package
Name      Version ID          Fix Versions
-----  -----
flask     1.0      PYSEC-2023-62  2.2.5,2.3.2
(myenv) laurencemoroney@Laurences-MacBook-Pro first_virtual %
```

# Understanding a specific security issue

**Prompt:** When I run pip-audit, I get a warning PYSEC-2023-62 in flask 1.0 – can you explain this?



When I run pip-audit, I get a warning PYSEC-2023-62 in flask 1.0 – can you explain this?



Searched 6 sites ▾

The warning you received, **PYSEC-2023-62**, refers to a vulnerability in Flask (versions prior to 2.2.5 and between 2.3.0 and 2.3.2) that exposes the risk of **session cookies being inadvertently shared across users**. This happens under specific conditions where an application:

1. Is hosted behind a caching proxy that doesn't strip or ignore cookies.
2. Has `session.permanent = True` set.
3. Does not modify the session during a request.
4. Uses `SESSION_REFRESH_EACH_REQUEST`, which is enabled by default.
5. Fails to set appropriate `Cache-Control` headers to prevent caching.

The issue arises because affected Flask versions do not set the `Vary: Cookie` header when refreshing the session, potentially leading to one user's session cookie being cached and sent to another user. This vulnerability can lead to information exposure and compromise user privacy.

The recommended fix is to upgrade Flask to **version 2.2.5 or 2.3.2** or higher to resolve this issue  
(OSV Dev)(GitHub)(GitHub)(VulnHub).



Message ChatGPT



ChatGPT can make mistakes. Check important info.



How can I fix the flask vulnerability to make my code secure?



To fix the **Flask vulnerability (PYSEC-2023-62)** and make your code more secure, follow these steps:

1. **Upgrade Flask to a secure version:** The vulnerability is resolved in Flask versions 2.2.5 and 2.3.2. You should update your `Flask` dependency to one of these or a newer version:

bash

Copy code

```
pip install Flask>=2.3.2
```

2. **Ensure proper session handling:** The vulnerability arises when `session.permanent = True` is set without sufficient protection in caching headers. To mitigate risks:

- Set `Cache-Control` headers to avoid caching of sensitive user data. You can do this globally by configuring Flask or adding the header in specific routes:

python

Copy code

```
@app.after_request
def add_header(response):
    response.headers['Cache-Control'] = 'no-store, no-cache, must-revalidate'
    return response
```



Message ChatGPT



ChatGPT can make mistakes. Check important info.



# Performing a security audit of your packages

**Prompt:** I am working on a simple web application in Python. Below are the dependencies currently listed for that project. Are there any known security issues for these libraries?

```
blinker==1.8.2
    # via flask
certifi==2024.6.2
    # via requests
charset-normalizer==3.3.2
    # via requests
click==8.1.7
    # via flask
flask==3.0.3
    # via -r requirements.in
idna==3.7
    # via requests
itsdangerous==2.2.0
    # via flask
...
...
```



GPT-4



As an AI model, I don't have real-time access to databases or the ability to perform live checks against security databases. However, I can guide you on how to check for vulnerabilities in your Python packages.

You can use tools like `**safety**` or `**pip-audit**` (when it becomes available) to check for known security vulnerabilities in Python packages. Here's how you can use `**safety**`:

1. Install `**safety**` if you haven't done so already:

```
bash
pip install safety
```

2. Run `**safety**` to check your requirements file for known security vulnerabilities:

```
bash
safety check -r requirements.txt
```

This will output a list of any installed packages that have known security vulnerabilities, along with information about the vulnerability and the affected versions.

Remember that these tools can only check for known vulnerabilities that have been reported and added to their databases. They can't guarantee that a package is completely free of security issues, especially newly discovered or unreported ones. Always follow best practices for secure coding to minimize the risk of security issues.



GPT-4o



To determine if there are any known security issues with the libraries listed, it's essential to check the latest security advisories and vulnerability databases. Here are some steps and resources you can use to verify the security status of these dependencies:

1. **Python Package Index (PyPI) Advisory Database:** PyPI provides security advisories for packages. You can check each package individually on the PyPI website.
2. **GitHub Security Advisories:** If your project is hosted on GitHub, you can use the GitHub Dependabot alerts to check for vulnerabilities in your dependencies.
3. **Common Vulnerabilities and Exposures (CVE) Database:** The CVE database is a comprehensive list of publicly disclosed cybersecurity vulnerabilities.
4. **Snyk:** Snyk is a tool that can scan your dependencies for known vulnerabilities.
5. **OWASP Dependency-Check:** This is another tool that can be used to identify known vulnerabilities in project dependencies.

Here is a quick check of the listed dependencies as of today:

1. **blinker==1.8.2**
  - No known security issues have been reported for this version.
2. **certifi==2024.6.2**
  - Certifi is a package that provides Mozilla's CA Bundle. There are no known security issues for this version.
3. **charset-normalizer==3.3.2**
  - No known security issues have been reported for this version.

# LLMs and dependency security

- LLMs are limited by the data they have access to
  - Know the cutoff date of your model's training data
  - Know whether your model can search the web

# LLMs and dependency security

- LLMs are limited by the data they have access to
  - Know the cutoff date of your model's training data
  - Know whether your model can search the web
- Use LLMs in parallel with other tools
  - Don't rely exclusively on LLMs to **identify** vulnerabilities
  - LLMs can be very useful in **resolving** vulnerabilities



DeepLearning.AI

# Dependency Management

---

## Dependency management in other languages

# LLMs and Dependencies

## Strengths

- Brainstorm libraries and packages to use for your project
- Learn more about a dependency
- Identify dependency conflicts
- Suggest solutions to issues with dependencies

## Weaknesses

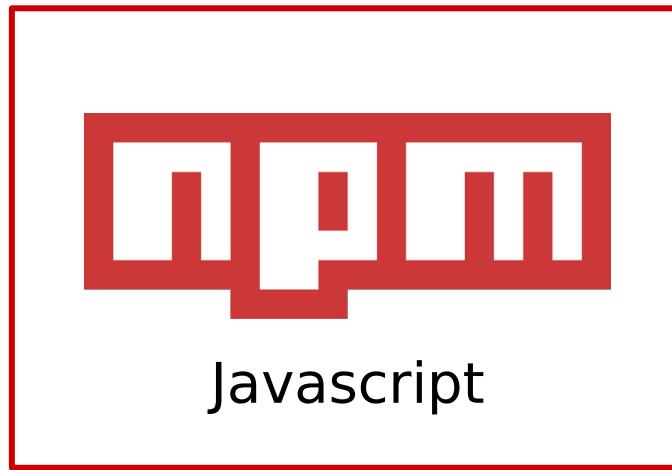
- LLMs may not know about dependency changes past their training date (some models do search the web)
- May be less helpful or accurate with obscure libraries that are rarely in their training data

# Managing Dependencies in Different Languages



Poetry

Python



nuget

C#

```
{  
  "name": "chatgpt-clone",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\"$Error: no test specified\\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "dotenv": "^16.4.5",  
    "express": "^4.19.2",  
    "openai": "^4.52.3",  
    "socket.io": "^4.7.5"  
  }  
}
```

```
{  
  "name": "chatgpt-clone",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\"$Error: no test specified\\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "dotenv": "^16.4.5",  
    "express": "^4.19.2",  
    "openai": "^4.52.3",  
    "socket.io": "^4.7.5"  
  }  
}
```

# Auditing Javascript packages with npm

**Prompt:** How can I check the following package.json file for known dependency issues?

```
{  
  "name": "chatgpt-clone",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "dotenv": "^16.4.5",  
    "express": "^4.19.2",  
    "openai": "^4.52.3",  
    "socket.io": "^4.7.5"  
  }  
}
```

> npm audit

> npm outdated

```
[laurence@Laurences-Mini chatgpt-clone % npm outdated
Package  Current  Wanted  Latest  Location          Depended by
openai    4.52.3   4.52.4  4.52.4  node_modules/openai  chatgpt-clone
laurence@Laurences-Mini chatgpt-clone % ]
```

> npm outdated

```
[laurence@Laurences-Mini chatgpt-clone % npm outdated
Package Current Wanted Latest Location      Depended by
openai  4.52.3  4.52.4  4.52.4  node_modules/openai  chatgpt-clone
laurence@Laurences-Mini chatgpt-clone % ]
```

> npm install openai@latest



DeepLearning.AI

# Team Software Engineering with AI

---

## Course conclusion



DeepLearning.AI

# Testing and Debugging

---

Analyzing code for  
security vulnerabilities