

lab_answer

April 3, 2026

1 Lab 15 — OpenAI GPT Series (ANSWERS)

Explore the OpenAI API, compare GPT-4o with Claude, and use GPT vision.

```
[ ]: # Setup
import time
import json
from dotenv import load_dotenv
from openai import OpenAI
import anthropic

load_dotenv()
openai_client = OpenAI()
claude_client = anthropic.Anthropic()

GPT_MODEL = "gpt-4o"
CLAUDE_MODEL = "claude-sonnet-4-20250514"
print("Setup complete - both clients ready.")
```

1.1 Exercise 1 — GPT-4o Basics

```
[ ]: prompt = "Explain the difference between a list and a tuple in Python. Give_
      ↪examples."

# GPT-4o call
gpt_response = openai_client.chat.completions.create(
    model=GPT_MODEL,
    messages=[{"role": "user", "content": prompt}],
    max_tokens=512,
)

print("=== GPT-4o Response ===")
print(gpt_response.choices[0].message.content)
print(f"\nTokens - prompt: {gpt_response.usage.prompt_tokens}, "
      f"completion: {gpt_response.usage.completion_tokens}")
```

```

# Claude call for comparison
claude_response = claude_client.messages.create(
    model=CLAUDE_MODEL,
    messages=[{"role": "user", "content": prompt}],
    max_tokens=512,
)

print("\n=== Claude Response ===")
print(claude_response.content[0].text)
print(f"\nTokens - input: {claude_response.usage.input_tokens}, "
      f"output: {claude_response.usage.output_tokens}")

# Structure comparison
print("\n=== Response Structure Comparison ===")
print(f"OpenAI: response.choices[0].message.content")
print(f"Anthropic: response.content[0].text")
print(f"OpenAI usage: prompt_tokens / completion_tokens")
print(f"Anthropic usage: input_tokens / output_tokens")

```

1.2 Exercise 2 — Side-by-Side Comparison

```

[ ]: TEST_PROMPTS = [
    "Write a Python function to find the longest palindrome in a string.",
    "Write a haiku about artificial intelligence.",
    "What causes tides on Earth? Explain in 3 sentences.",
    "A farmer has 17 sheep. All but 9 die. How many are left? Explain your_
↪ reasoning.",
    "Summarise the key ideas of object-oriented programming in 5 bullet points.
↪",
]

results = []

for i, prompt in enumerate(TEST_PROMPTS):
    print(f"Running prompt {i+1}/5...")

    # GPT-4o
    t0 = time.time()
    gpt_r = openai_client.chat.completions.create(
        model=GPT_MODEL,
        messages=[{"role": "user", "content": prompt}],
        max_tokens=300,
    )
    gpt_time = time.time() - t0

```

```

# Claude
t0 = time.time()
claude_r = claude_client.messages.create(
    model=CLAUDE_MODEL,
    messages=[{"role": "user", "content": prompt}],
    max_tokens=300,
)
claude_time = time.time() - t0

results.append({
    "prompt": prompt[:50] + "...",
    "gpt_text": gpt_r.choices[0].message.content[:150],
    "gpt_tokens": gpt_r.usage.prompt_tokens + gpt_r.usage.completion_tokens,
    "gpt_time": gpt_time,
    "claude_text": claude_r.content[0].text[:150],
    "claude_tokens": claude_r.usage.input_tokens + claude_r.usage.
↪output_tokens,
    "claude_time": claude_time,
})

# Display comparison table
print(f"\n{'Prompt':<52} {'GPT Tok':>8} {'GPT Time':>9} {'Claude Tok':>11}␣
↪{'Claude Time':>12}")
print("-" * 95)
for r in results:
    print(f"{r['prompt']:<52} {r['gpt_tokens']:>8} {r['gpt_time']:>8.2f}s␣
↪{r['claude_tokens']:>11} {r['claude_time']:>11.2f}s")

# Show full responses for first prompt
print(f"\n{'='*80}")
print(f"Full comparison for: {TEST_PROMPTS[0][:60]}")
print(f"\n--- GPT-4o ---\n{results[0]['gpt_text']}")
print(f"\n--- Claude ---\n{results[0]['claude_text']}")

```

1.3 Exercise 3 — GPT-4o Vision

```

[ ]: SAMPLE_IMAGE_URL = "https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/
↪Cat03.jpg/1200px-Cat03.jpg"

# Basic description
response = openai_client.chat.completions.create(
    model=GPT_MODEL,
    messages=[{
        "role": "user",
        "content": [

```

```

        {"type": "text", "text": "Describe this image in detail."},
        {"type": "image_url", "image_url": {"url": SAMPLE_IMAGE_URL,
↪ "detail": "high"}}},
    ],
    },
    max_tokens=300,
)
print("=== Image Description (high detail) ===")
print(response.choices[0].message.content)
print(f"Tokens: {response.usage.prompt_tokens} input, {response.usage.
↪ completion_tokens} output")

# Low detail comparison
response_low = openai_client.chat.completions.create(
    model=GPT_MODEL,
    messages=[{
        "role": "user",
        "content": [
            {"type": "text", "text": "Describe this image in detail."},
            {"type": "image_url", "image_url": {"url": SAMPLE_IMAGE_URL,
↪ "detail": "low"}}},
    ],
    },
    max_tokens=300,
)
print(f"\n=== Image Description (low detail) ===")
print(response_low.choices[0].message.content)
print(f"Tokens: {response_low.usage.prompt_tokens} input, {response_low.usage.
↪ completion_tokens} output")

print(f"\n=== Token Cost Comparison ===")
print(f"High detail input tokens: {response.usage.prompt_tokens}")
print(f"Low detail input tokens: {response_low.usage.prompt_tokens}")
print(f"Difference: {response.usage.prompt_tokens - response_low.usage.
↪ prompt_tokens} tokens")

```