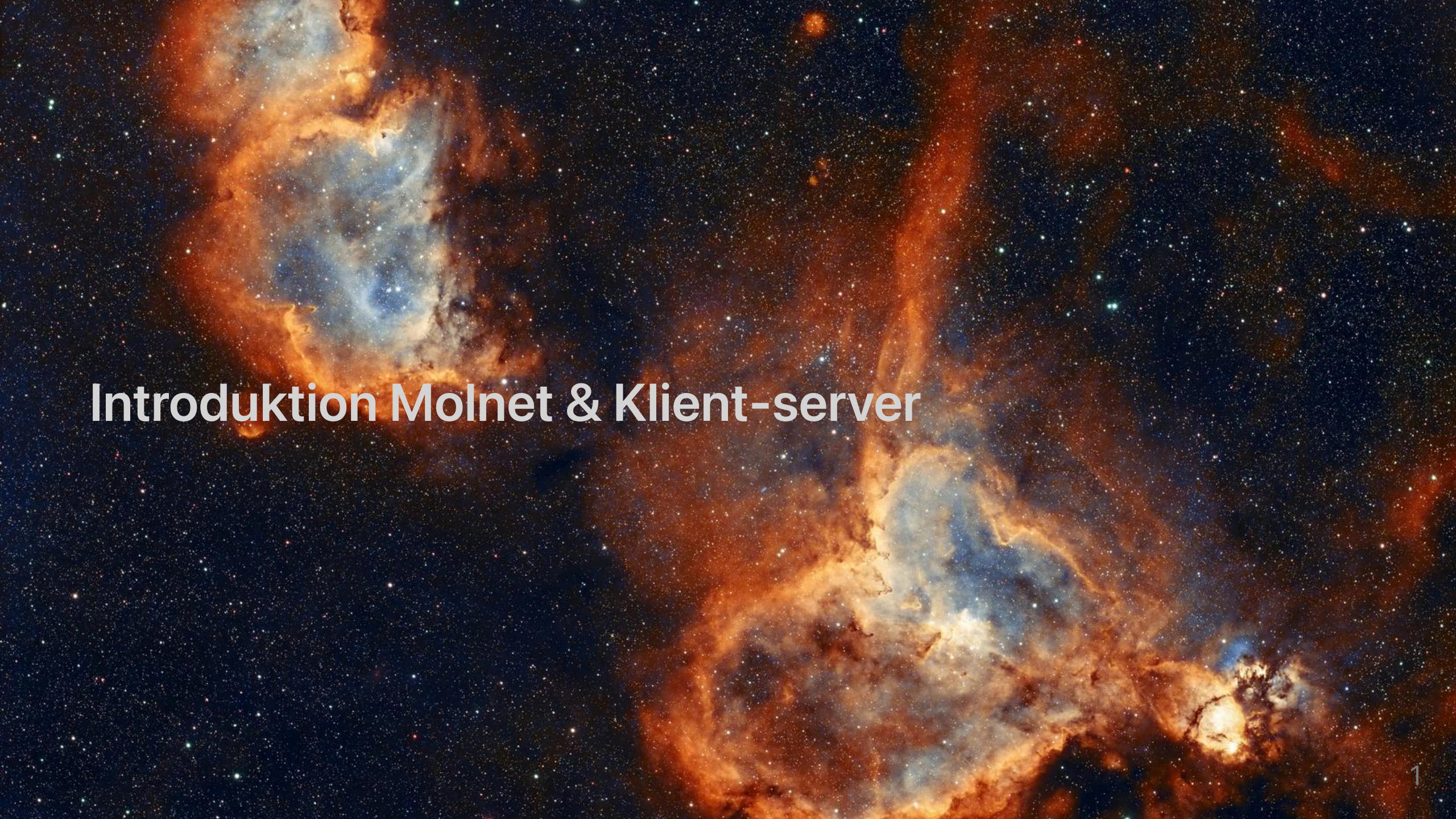

Introduktion Molnet & Klient-server

Dagens agenda

- Molnet
- Klient-server
- API och API-anrop
- Stora språkmodeller

Dagens nya ord:

- moln, klient-server, url, api, json, llm, http, statuskod,

☁️ Vad är "molnet"?

- Molnet = extern hårddisk (eller *server*) på nätet som vi kan lagra data på och komma åt när och var som helst
- Istället för att lagra data på egen dator görs det via molnet
- Bra vid viss typ av säkerhetskopiering och delning av stora filer med andra
- Eventuellt olämpligt om man helt ersätter lokal lagring med molnlagring

Molntjänster - exempel

- Lagring
  - Google Drive, Dropbox, Box, icloud
- Streaming
  - Spotify, Netflix
- Beräkning
  - Azure, Google Cloud, lokala beräkningskluster

💻 Klient-server

- Klient = en användarenhet, ex.vis en webbläsare eller app
- Server = (kraftfull) dator som levererar tjänster eller data på begäran
- Klienten *ber alltså om något*, ex.vis *"ge mig mina mail"*, och servern svarar med efterfrågad information

🔗 Vad är ett API/API-anrop?

- API = Application Programming Interface
  - Ett "gränssnitt" som möjliggör kommunikation mellan program
- Ett API-anrop är i princip att klienten skickar en fråga till servern, ex.vis *"Hur är vädret i Växjö"*
- Servern svarar därefter med relevant data
- API:er används ofta för att hämta/lämna data från molntjänster

API: klienter och servrar

API är en form av regelbok eller gränssnitt som förklarar:

- Vad klienten kan/"får" fråga om
- Hur den ska ställa sina frågor
- Vad servern kommer svara med

Hur hänger ovanstående ihop?

1. En app på din mobil vill hämta nyheter
2. Den gör ett API-anrop till en server i molnet
3. Servern letar upp rätt data och skickar dig ett svar
4. Appen visar nyheterna för dig

Annat (lite löjligt) exempel

Restaurangbesök

- Du (klienten) sätter dig vid bordet och läser menyn (det som går att beställa/hämta)
- Du berättar för servitören (API:et) vad du vill ha
- Servitören går till köket (servern), hämtar maten (data) och levererar till dig

Poängen är att du inte behöver veta hur köket fungerar (hur servern lagrar eller bearbetar data). Du bara *ber om något* genom API:et, och *får ett svar*.

Exempel på klient-server/API-anrop

Hämta slumpmässig kattbild med ett `Python` -skript

```python
import requests

url = "https://api.thecatapi.com/v1/images/search"
response = requests.get(url)

if response.status_code == 200:
  data = response.json()
  print("Här är en kattbild:", data[0]["url"])
else:
  print("Något gick fel:", response.status_code)
```

Vad händer här?

- `requests.get(...)` skickar förfrågan från klienten (vårt program)
- API:et tar emot förfrågan och skickar den till servern (där kattbilderna finns)
- Servern väljer slumpmässig bild och returnerar ett svar
- Svaret innehåller bildens URL, som man kan klicka på för att se katten

Man kan också följa länken direkt https://api.thecatapi.com/v1/images/search för att se hur ett API-svar ser ut "rått".

200?

```
if response.status_code == 200:
```

Finns olika statuskoder:

Kod	Betydelse	Kommentar
200	✅ OK	Allt gick bra – vi fick ett svar
404	❌ Not Found	Resursen finns inte (fel URL?)
500	🔥 Server Error	Något gick sönder på serversidan
403	⛔ Forbidden	Du har inte rättigheter (t.ex. ingen API-nyckel)
401	🔑 Unauthorized	Du måste logga in eller använda nyckel

data?

`print(data)` ger

```
[{'id': 'c2t', 'url': 'https://cdn2.thecatapi.com/images/c2t.jpg', 'width': 900, 'height': 1200}]
```

`print(data[0])` ger

```
{'id': 'c2t', 'url': 'https://cdn2.thecatapi.com/images/c2t.jpg', 'width': 900, 'height': 1200}
```

- Tar alltså bara bort hakparentesen
- ...och returnerar en Python-dictionary (även kallat associativt fält eller en hash-tabell)
  - "dict": varje *värde* har en *nyckel* (mer kring detta strax)

Och `print(data[0]["url"])` ger oss själva länken till bilden

```
https://cdn2.thecatapi.com/images/c2t.jpg
```

Plocka mängder av kattbilder på nolltid

Annat kattkollage med samma anrop

HTTP

- Ovan katt-exempel är ett typexempel på en HTTP-förfrågan
- HTTP-förfrågan är en av de vanligaste *sätten* att göra en API-förfrågan
- Vi skickade ovan HTTP-förfrågan med `get` och fick en URL (Uniform Resource Locator = "länk")
- Servern svarade (som ofta) med JSON-data

Begrepp	Vad det är	Exempel
API	Allmänt gränssnitt mellan system	En Python-funktion eller webbtjänst
HTTP-förfrågan	Specifik teknisk kommunikation	Webbläsare laddar en webbsida
Web API	Ett API man når via HTTP	`GET https://api.weather.com/...`

Jaha, vad är då JSON? 😩

- JSON står för JavaScript Object Notation
- Format för att strukturera data på ett enkelt och lättläst sätt
- Används ofta för att skicka data mellan server och klient, ex.vis i ett API-svar

Exempel på JSON 🐈

```
{
  "name": "Truckis",
  "age": 13,
  "colors": ["white", "gray", "black"],
  "owner": {
    "name": "Alex",
    "location": "Växjö"
  }
}
```

JSON-exempel, forts

- `"name": "Truckis"` – Nyckeln är `"name"`, värdet är `"Truckis"`.
- `"age": 13` – Nyckeln är `"age"`, värdet är `13`.
- `"colors": ["white", "gray", "black"]` – Nyckeln är `"colors"`, och värdet är en lista med tre färger.
- `"owner": {...}` – Här är värdet ett objekt som innehåller ytterligare nyckel-värde-par, som beskriver kattens ägare.

(Litet sidospår: Skillnaden mellan `dict` och `json`)

Aspekt	Python dict	JSON
Vad det är	En datastruktur i Python	Ett textformat (sträng)
Syntax	Python-syntax	Striktare (t.ex. bara dubbla citationstecken)
Användning	I Python-program	För nätverk, API, filöverföring
Typ	Python-objekt	Sträng
Exempel	{"namn": "Alex", "ålder": 42}	'{"namn": "Alex", "ålder": 42}'

Samarbete mellan molnet och klient-server

- Lagra och hämta data via ett API
- Exempel:
  - Google Calendar API
  - Är molnbaserad tjänst där användare kan lagra sina kalenderhändelser.
  - Lägg till ny händelse via deras API
  - Flöde:
    - Klienten (din app eller hemsida) skickar HTTP-förfrågan till GC API
    - Förfrågan kan vara att lägga till händelse eller uppdatera en befintlig
    - GC-API:et (som är en server i molnet) svarar med JSON-data

Hur detta kan se ut under huven

```python
import requests

url = "https://www.googleapis.com/calendar/v3/calendars/primary/events"
headers = {
    'Authorization': 'Bearer YOUR_ACCESS_TOKEN',
}
data = {
    'summary': 'Möte med XX',
    'start': {'dateTime': '2025-04-11T09:00:00', 'timeZone': 'Europe/Stockholm'},
    'end': {'dateTime': '2025-04-11T10:00:00', 'timeZone': 'Europe/Stockholm'}
}

response = requests.post(url, headers=headers, json=data)
if response.status_code == 200:
    print("Event added successfully!")
else:
    print("Error:", response.status_code)
```

Övrigt, molnet

- Många populära appar i mobilen/datorn använder molnbaserade tjänster för att synkronisera data mellan olika enheter
- Kan komma åt sin data var som helst, på vilken enhet som helst (förutsatt www)
  - Tillgänglighet, säkerhet, skalbarhet.
  - Tjänster som kan användas globalt utan att behöva egna servrar.
- Data är tryggt i molnet (men jag lagrar fortf foton enbart på HD + BackupHD ändå 🙈)
  - Sekretess och säkerhet
  - Beroende av uppkoppling

Stora språkmodeller

Kort om stora språkmodeller (LLMs) och hur dessa kan anropas och skräddarsys med Python

- En LLM är en avancerad matematisk funktion som förutspår vilket ord som kommer härnäst för en given text
- Blir allt bättre på datorkod-generering
- Kan skräddarsys via ex.vis API-anrop
- Finns numera förnuftiga lokalt körbara LLMs om man har känslig data, etc.

LLM och kodning

- Stora språkmodeller blir allt bättre på att programmera
- När man är bekväm med grunderna kan man med fördel ta hjälp av ex.vis ChatGPT
- Även en nybörjarkodare kan skapa ganska avancerade program med LLMs
- Men! - man bör/måste ju kunna förstå, ta ansvar för, och justera den kod LLM:en genererar

LLM och kodning

- Kortare kodsnuttar kan genereras direkt i en LLMs webinterface
- Numera finns även möjligheter att skapa mer komplex kod-arkitektur med LLMs
- Siffror varierar, men färska siffror säger att ca 50% kod idag genereras med LLMs
- Höjer nivån för nybörjare, medan erfarna utvecklare kan spara tid för "enklare" och/eller "tråkig" kodgenerering

Enkelt exempel

Primtalsgenereringskod

Prompt: *Vänligen ge mig en Python-kod som genererar alla primtal mellan 0 och 100. Lättbegriplig kod*

(LLM-utmatning på nästa sida)

```python
# Funktion för att kontrollera om ett tal är ett primtal
def är_primtal(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# Lista för att spara primtalen
primtal = []

# Gå igenom alla tal från 0 till 100
for tal in range(101):
    if är_primtal(tal):
        primtal.append(tal)

# Skriv ut resultatet
print("Primtal mellan 0 och 100:")
print(primtal)
```

Utmatning från genererat skript:

```
Primtal mellan 0 och 100: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Ny prompt: *Vänligen ge mig en Python-kod som genererar alla primtal mellan 0 och 100. Så kort kod som möjligt.*

```
print([n for n in range(2, 101) if all(n % d != 0 for d in range(2, int(n**0.5)+1))])
```

Utmatning:

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Bugg-fix med GPT

Kod (inkl. medvetet fel):

```
print([n for n in range(2, 101) if All(n % d != 0 for d in range(2, int(n**0.5)+1))])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 print([n for n in range(2, 101) if All(n % d != 0 for d in range(2, int(n**0.5)+1))])

Cell In[2], line 1
----> 1 print([n for n in range(2, 101) if All(n % d != 0 for d in range(2, int(n**0.5)+1))])

NameError: name 'All' is not defined
```

Ge GPT kod + felmeddelande

Kod: ...

Error: ...

GPT-utmatning:

Bra att du dubbelkollar! Felet beror på att Python-funktionen heter all (med liten bokstav), inte All (med stor bokstav).

Här är den korrekta koden:

```
print([n for n in range(2, 101) if all(n % d != 0 for d in range(2, int(n**0.5)+1))])
```

✅ all(...) returnerar True om alla villkor i generatorn är sanna.

Testa den — nu ska det fungera perfekt! 🧠✨

Inom snar framtid (redan nu?)

- LLMs förväntas kunna skriva ML-kod samt *exekvera* och utvärdera/förbättra denna
  - Läsa nya forskningsartiklar inom ML och implementera metodologin för punkt 1 ovan
  - Föreslå förbättringar/effektiviseringar till sig själv
  - Detta i en etablerad programutvecklingsarkitektur (dvs i en viss kodkatalogstruktur)
- Absoluta merparten av all datorkod kommer genereras av AI (tyvärr?)
- Allt högre kvalitet på genererad kod

GPT via API

- Varför?
  - Analysera en större mängd textfiler via `for`-loop eller motsv.
  - Mer kontroll på LLM:ens systeminställningar kring hur den genererade texten ska se ut.
  - Integration med andra verktyg och databaser (Pandas, Excel, etc.)

Exempel

```
import json
from openai import OpenAI

with open("config.json", "r", encoding="utf-8") as f:
    api_key = json.load(f).get("openai_api_key")

client = OpenAI(api_key=api_key)
system_prompt = "Svara så att en nioåring förstår."
user_question = "Varför är himlen blå?"
response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": user_question}
    ],
    temperature=0.1  # Högre temp -> mer slumpmässig/kreativ
)
print("\nSvar från GPT:\n")
print(response.choices[0].message.content.strip())
```

Exempel, forts.

`.json`-filen ska se ut enligt följande

```
# json-fil med namn config.json och API-nyckel. Ska se ut så här:
{
    "openai_api_key": "sk-234dummyAPInyckel34jlkj234"
  }
```

Klistra in din API-nyckel mellan citations-tecknen.

Genererat svar från modellen gpt-4o-mini

"Himlen är blå för att när solens ljus kommer ner till jorden, så träffar det luften. Ljuset består av många olika färger, och blått ljus sprids mer än de andra färgerna när det träffar luftens små partiklar. Så när vi tittar upp på himlen, ser vi mest det blå ljuset som sprids runt. Det är därför himlen ser blå ut!"

Svar som genereras efter att ha ändrat system prompten till `system_prompt = "Svara till en fysik-kunnig person."`:

"Himlen ser blå ut på grund av ett fenomen som kallas Rayleigh-spridning. När solens ljus passerar genom atmosfären, interagerar det med luftmolekyler och små partiklar. Solens ljus består av olika färger, som alla har olika våglängder. Blått ljus har en kortare våglängd än rött ljus. Eftersom kortvågigt ljus (som blått) sprids mer effektivt än långvågigt ljus (som rött) när det träffar molekyler i atmosfären, sprids det blå ljuset i alla riktningar. När vi tittar upp på himlen ser vi därför mer av det blå ljuset än av de andra färgerna, vilket ger himlen dess karakteristiska blå färg. Under solnedgång och soluppgång, när ljuset passerar genom en tjockare del av atmosfären, sprids det blå ljuset ännu mer, och de längre våglängderna (som rött och orange) blir mer framträdande, vilket ger upphov till de vackra färgerna vi ser vid dessa tider."

Kort demo, lokal LLM

- Finns numera mindre LLMs som kan köras lokalt på moderna laptops
- Ex.vis Llama, Deepseek, etc.
    - Kan köras med hjälpprogram: Ollama (ollama.com <- där finns även instruktioner)
- Bra om man jobbar med sekretessbelagd data och liknande

Praktiska tillämpningar LLM-API:er

- Analysera transkriberade telefonsamtal för statistik och kvalitetssäkring
- Översätta produktbeskrivningar till många andra språk i ex.vis ett Excel-ark
- Automatgenerera delar av legala/patenttekniska dokument baserat på avancerade systemmeddelanden/prompter
- Skapa egen chatbot (RAG) som är expert på ett smalt område
- Klassificera fakturarader