In [1]:
```
!pip install -q -U google-generativeai
```

In [2]:
```
import os
os.environ['GEMINI_API_KEY']="AIzaSyDfaWjommA_0hSTl33FHeF-vgXWrsWCSN0"
```

In [3]:
```
import google.generativeai as genai
genai.configure(api_key=os.environ['GEMINI_API_KEY'])
```

In [4]:
```
model=genai.GenerativeModel('gemini-2.5-flash-preview-04-17')
```

In [5]:
```
response = model.generate_content("WHAT WOULD BE THE FUTURE SCOPE OF GENERATIVE AI
```

In [6]:
```
print(response.text)
```

Okay, let's look at the future scope of Generative AI *specifically as it unfolded and was perceived* during 2023. 2023 was arguably the year Generative AI went mainstream, moving from research labs to widespread public and enterprise awareness.

Here's a breakdown of the key areas and perceived future scope during that year:

1. **Explosive Growth in Application & Adoption:**
    * **Wider Accessibility:** Tools became more user-friendly and integrated into existing platforms (e.g., Microsoft Copilot in Office, Adobe Firefly in Creative Cloud).
    * **Enterprise Adoption:** Businesses across various sectors (marketing, customer service, software development, finance) began piloting and implementing GenAI for tasks like content generation, code assistance, data analysis, and automation.
    * **Individual Productivity:** Millions started using tools like ChatGPT, Midjourney, and others for writing, brainstorming, coding help, image creation, etc., transforming personal workflows.

2. **Expansion of Capabilities:**
    * **Multimodality:** Models started moving beyond just text to handle and generate combinations of text, images, audio, and even video (though video was still nascent). The ability to understand and create across different data types opened up vast new possibilities.
    * **Improved Quality & Coherence:** Models became better at generating longer, more coherent, and contextually relevant content (text, code, images) with fewer obvious errors or nonsensical outputs.
    * **Increased Speed & Efficiency:** While still computationally expensive, inference times for many models decreased, making real-time interactions more feasible.

3. **Integration into Existing Workflows:**
    * A major focus was integrating GenAI *into* the tools people already used (search engines like Bing/Google, productivity suites, design software, development environments). The "future scope" in 2023 wasn't just standalone AI tools, but AI as an embedded feature.

4. **Diversification of Use Cases:**
    * Beyond initial content creation, GenAI's scope expanded to areas like:
        * **Personalization:** Generating highly tailored content or experiences.
        * **Simulation & Modeling:** Creating synthetic data or simulating complex scenarios.
        * **Research & Discovery:** Summarizing vast amounts of information, generating hypotheses.
        * **Education:** Creating personalized learning materials, tutoring bots.
        * **Gaming & Entertainment:** Generating game assets, story elements, interactive experiences.

5. **Focus on Fine-tuning & Customization:**
    * The ability for organizations to fine-tune models on their specific data became a significant area of development, promising GenAI solutions tailored to internal processes, knowledge bases, and brand voices.

6. **Significant Investment & Competition:**
    * 2023 saw massive investment from tech giants and venture capital into GenAI startups and research. This competitive landscape accelerated development and pushed the boundaries of what was possible, indicating a belief in its long-term scope.

**However, the "future scope" in 2023 also included significant challenges and areas

of focus:**

*   **Ethics & Governance:** Discussions around bias, fairness, responsible deployment, and preventing misuse (deepfakes, misinformation) became critical and were actively addressed by researchers, companies, and policymakers.
*   **Regulation:** Governments globally began exploring regulations for GenAI, acknowledging its potential impact on society and the economy.
*   **Copyright and Ownership:** Questions around who owns AI-generated content and the legality of training data became contentious issues needing resolution.
*   **Job Market Impact:** The potential for disruption and the need for reskilling/upskilling were major talking points.
*   **Compute & Cost:** The expense and energy requirements for training and running large models remained a significant factor limiting broader access and certain applications.
*   **Reliability & "Hallucinations":** While improving, models could still generate incorrect or nonsensical information, highlighting the need for human oversight and verification.

In summary, the "future scope" of Generative AI as envisioned and enacted in 2023 was one of:

*   **Rapid, widespread adoption and integration.**
*   **Expanding capabilities beyond text to multimodal generation.**
*   **Moving from novel tools to essential productivity enhancers and business solutions.**
*   **Significant innovation driven by intense competition and investment.**
*   **Simultaneously, a crucial period of grappling with the ethical, societal, and technical challenges inherent in this powerful technology.**

2023 was less about distant predictions and more about the *immediate* explosion of practical applications and the urgent need to understand and shape its impact.

In [7]:
```
import pathlib
import textwrap
```

In [8]:
```
import google.generativeai as genai
```

In [9]:
```
from IPython.display import display
from IPython.display import Markdown
```

In [10]:
```
def to_markdown(text):
    text = text.replace('.','*')
    return Markdown(textwrap.indent(text, '>',predicate= lambda _: True))
```

In [11]:
```
for m in genai.list_models():
    if 'generateContent' in m.supported_generation_methods:
        print(m.name)
```

```
models/gemini-1.0-pro-vision-latest
models/gemini-pro-vision
models/gemini-1.5-pro-latest
models/gemini-1.5-pro-001
models/gemini-1.5-pro-002
models/gemini-1.5-pro
models/gemini-1.5-flash-latest
models/gemini-1.5-flash-001
models/gemini-1.5-flash-001-tuning
models/gemini-1.5-flash
models/gemini-1.5-flash-002
models/gemini-1.5-flash-8b
models/gemini-1.5-flash-8b-001
models/gemini-1.5-flash-8b-latest
models/gemini-1.5-flash-8b-exp-0827
models/gemini-1.5-flash-8b-exp-0924
models/gemini-2.5-pro-exp-03-25
models/gemini-2.5-pro-preview-03-25
models/gemini-2.5-flash-preview-04-17
models/gemini-2.0-flash-exp
models/gemini-2.0-flash
models/gemini-2.0-flash-001
models/gemini-2.0-flash-exp-image-generation
models/gemini-2.0-flash-lite-001
models/gemini-2.0-flash-lite
models/gemini-2.0-flash-lite-preview-02-05
models/gemini-2.0-flash-lite-preview
models/gemini-2.0-pro-exp
models/gemini-2.0-pro-exp-02-05
models/gemini-exp-1206
models/gemini-2.0-flash-thinking-exp-01-21
models/gemini-2.0-flash-thinking-exp
models/gemini-2.0-flash-thinking-exp-1219
models/learnlm-1.5-pro-experimental
models/learnlm-2.0-flash-experimental
models/gemma-3-1b-it
models/gemma-3-4b-it
models/gemma-3-12b-it
models/gemma-3-27b-it
```

In [12]:
```
model = genai.GenerativeModel('gemini-2.5-flash-preview-04-17')
#model = genai.GenerativeModel('gemini-2.5-pro-exp-03-25')
#model = genai.GenerativeModel('gemini-2.0-flash-exp-image-generation')
```

In [13]:
```
%%time
response = model.generate_content("create a table for summarize to a kid about data
```

```
CPU times: total: 93.8 ms
Wall time: 21.5 s
```

In [14]:
```
to_markdown(response.text)
```

Out[14]:

Okay, that's a lot of big words! Let's break them down into simple ideas you can understand, like comparing different types of superhero powers or tools*

I can create the table for you right here, but I **can't directly save a file to your computer as a PDF*** That's something only you can do on your own computer because I don't have access to your files for safety reasons*

Here's how you can save it as a PDF yourself after I give you the table:

1* I will give you the information in a clear table format* 2* You will need to **copy** the entire table text I provide* 3* **Paste** it into a document program you have on your computer, like Microsoft Word, Google Docs (if you use Google Drive), or even a simple text editor like Notepad (though Word/Google Docs is better for tables)* 4* In that program, go to the **"File" menu*** 5* Look for options like **"Save As"** or **"Export"*** 6* When choosing the file type, select **"PDF"*** 7* Choose where you want to save it on your computer and click **"Save"***

Now, here is the simple table comparing those concepts for you:

| What is it? | Simple Explanation (for a kid!) | Think of it like*** | Why is it cool? | | :---------- | :---------- | :---------- | :---------- | | **Data Science** | Being a detective who looks at *lots* of clues (data) to find stories, patterns, or guess what might happen next* | A detective looking at all the evidence to solve a mystery* | Helps us understand things better and make smart choices! | | **Machine Learning (ML)** | Teaching computers to learn things by looking at lots of examples, instead of telling them every single step* | A robot learning to spot cats by looking at millions of cat pictures* | Computers get smarter over time, like recommending videos you like! | | **Natural Language Processing (NLP)** | Making computers understand and use human languages, like English or Spanish* | A translator helping people talk to each other, or talking to your smart speaker* | We can talk to computers and they can help us with words! | | **Deep Learning (DL)** | A *special* kind of Machine Learning that uses really big, layered "brains" inside the computer to learn super complex things* | Using a super-powered, layered brain to learn tricky stuff, like recognizing faces perfectly* | Great for understanding complicated things like images and sounds really well* | | **Neural Network (NN)** | The "brain" structure that Deep Learning uses! It's made of layers of connections, like a simplified version of your own brain* | A network of friends passing messages and learning secrets, layer by layer* | It's the powerful tool that makes Deep Learning possible! | | **Generative AI**

> **(GenAI)** | AI that can create *new* stuff, like drawing pictures, writing stories, or making music* | An artist who can draw anything you describe, or a writer who can make up new tales* | Can be super creative and help us make amazing new art or stories! | | **Agentic AI** | AI that can not only answer questions but can plan steps and *do* things to reach a goal, like booking a trip* | A personal assistant who can plan and *do* all the steps to get a task finished for you* | Helps automate complex tasks and get things done without needing constant instructions* | | **MCP (Master Control Program)** | **Wait! This one is different!** It's not a real scientific term like the others* It's a powerful computer character from a sci-fi movie called TRON! | Like a super-smart, sometimes bossy, computer character in a movie* | It's cool in the movie, but it's not a real field of AI science* | | **AGI (Artificial General Intelligence)** | A super-duper smart computer that is as smart and capable as a human across *many* different tasks* (We don't have this yet!) | A robot friend who is just as smart as a human and can learn and do anything* | (Explain it's future) Could solve huge problems, but it's something scientists are still dreaming about and working towards (and debating!)* |
>
> I hope this table helps make sense of those terms! Remember to copy and paste it into a document program to save it as a PDF on your computer*

In [15]:
```
for chunk in response:
    print(chunk.text)
```

Okay, that's a lot of big words! Let's break them down into simple ideas you can und
erstand, like comparing different types of superhero powers or tools.

I can create the table for you right here, but I **can't directly save a file to you
r computer as a PDF**. That's something only you can do on your own computer because
I don't have access to your files for safety reasons.

Here's how you can save it as a PDF yourself after I give you the table:

1. I will give you the information in a clear table format.
2. You will need to **copy** the entire table text I provide.
3. **Paste** it into a document program you have on your computer, like Microsoft W
ord, Google Docs (if you use Google Drive), or even a simple text editor like Notepa
d (though Word/Google Docs is better for tables).
4. In that program, go to the **"File" menu**.
5. Look for options like **"Save As"** or **"Export"**.
6. When choosing the file type, select **"PDF"**.
7. Choose where you want to save it on your computer and click **"Save"**.

Now, here is the simple table comparing those concepts for you:

| What is it? | Simple Explanation (for a kid!)
| Think of it like...
| Why is it cool?
|
| :---------- | :----------

Data Science	Being a detective who looks at *lots* of clues (data)
to find stories, patterns, or guess what might happen next.	A detective looking at
all the evidence to solve a mystery.	Helps us understand
things better and make smart choices!
Machine Learning (ML)	Teaching computers to learn things by looking at lots
of examples, instead of telling them every single step.	A robot learning to spot c
ats by looking at millions of cat pictures.	Computers get smarter o
ver time, like recommending videos you like!
Natural Language Processing (NLP)	Making computers understand and use human
languages, like English or Spanish.	A translator helping people ta
lk to each other, or talking to your smart speaker.	We can talk to computers an
d they can help us with words!
Deep Learning (DL)	A *special* kind of Machine Learning that uses really
big, layered "brains" inside the computer to learn super complex things.	Using a s
uper-powered, layered brain to learn tricky stuff, like recognizing faces perfectly.
| Great for understanding complicated things like images and sounds really well.
|
| **Neural Network (NN)** | The "brain" structure that Deep Learning uses! It's ma
de of layers of connections, like a simplified version of your own brain. | A networ
k of friends passing messages and learning secrets, layer by layer. | It's
the powerful tool that makes Deep Learning possible! |
| **Generative AI (GenAI)** | AI that can create *new* stuff, like drawing pictures,
writing stories, or making music. | An artist who can draw anything you
describe, or a writer who can make up new tales. | Can be super creative and help us
make amazing new art or stories! |

Agentic AI	AI that can not only answer questions but can plan steps and *do* things to reach a goal, like booking a trip.	A personal assistant who can plan and *do* all the steps to get a task finished for you.	Helps automate complex tasks and get things done without needing constant instructions.
MCP (Master Control Program)	**Wait! This one is different!** It's not a real scientific term like the others. It's a powerful computer character from a sci-fi movie called TRON!	Like a super-smart, sometimes bossy, computer character in a movie.	It's cool in the movie, but it's not a real field of AI science.
AGI (Artificial General Intelligence)	A super-duper smart computer that is as smart and capable as a human across *many* different tasks. (We don't have this yet!)	A robot friend who is just as smart as a human and can learn and do anything.	(Explain it's future) Could solve huge problems, but it's something scientists are still dreaming about and working towards (and debating!).

I hope this table helps make sense of those terms! Remember to copy and paste it into a document program to save it as a PDF on your computer.

TEXT MODEL WE BUILT USING DIFFERENNT GEMINI MODELS & LATEST MODELS

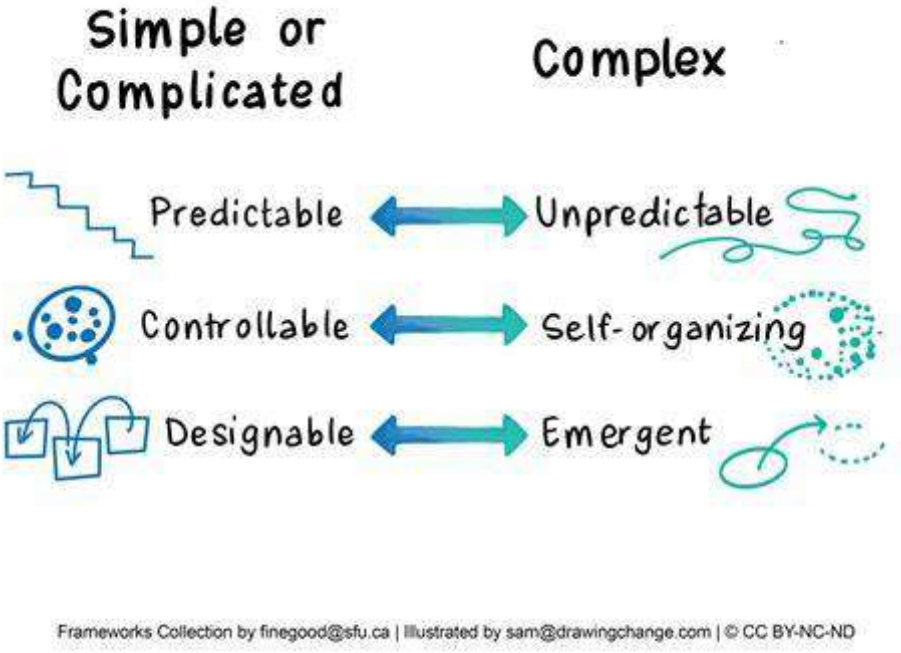
(IMAGE-TEXT GEN)

In [18]:
```
import PIL.Image

img = PIL.Image.open(r"C:\Users\akaft\OneDrive\Desktop\OIP.jpg")
```

In [19]:
```
img
```

Out[19]:



In [20]:
```
#model = genai.GenerativeModel('models/gemini-2.0-pro-exp-02-05')

#model = genai.GenerativeModel('models/gemini-2.5-pro-exp-03-25')

#model = genai.GenerativeModel('gemini-2.0-flash-exp-image-generation')

model = genai.GenerativeModel('gemini-2.5-flash-preview-04-17')
```

In [21]:
```
response = model.generate_content(img)
```

In [22]:
```
to_markdown(response.text)
```

Out[22]:

The image presents a comparison between "Simple or Complicated" systems and "Complex" systems*

Under "Simple or Complicated", the characteristics are listed as:

- Predictable (illustrated by a staircase icon)
- Controllable (illustrated by a circle with dots inside and outside)
- Designable (illustrated by three squares connected by lines, suggesting inputs/outputs)

Under "Complex", the characteristics are listed as:

- Unpredictable (illustrated by a wavy line with loops)
- Self-organizing (illustrated by a dotted circle with dots inside and outside, suggesting emergent pattern)
- Emergent (illustrated by a dotted line moving from a circle and curving, suggesting a new path)

Double-headed arrows connect the characteristics across the two columns, indicating a spectrum or contrast:

- Predictable is connected to Unpredictable
- Controllable is connected to Self-organizing
- Designable is connected to Emergent

At the bottom of the image, there is text that reads: "Frameworks Collection by finegood@sfu*ca* | *Illustrated by sam@drawingchange*com | © CC BY-NC-ND"*

In [23]:
```
import PIL.Image

img1 = PIL.Image.open(r"C:\vs_code_sk\horse_img.png")
img1
```

Out[23]:

In [24]:
```
response1 = model.generate_content(img1)

to_markdown(response1.text)
```

Out[24]:

> A majestic white horse is captured in motion, appearing to gallop through shallow water or mist* The horse is centered in the vertical frame, shown from the front as it strides forward* Its powerful white form stands out against a muted, dark blue-gray background* Its white mane and tail are dramatically swept back, suggesting speed and wind* Around the horse's legs, spray or mist rises from the water's surface, adding to the sense of dynamic movement* The bottom of the image shows the dark water reflecting the horse and the ambient light, creating ripples and subtle distortion* The lighting accentuates the horse's muscular physique and the texture of its flowing hair, giving the image an ethereal and powerful quality*

In [25]:
```
response = model.generate_content(["Write a script for 5sec instagram shoot to gain
response.resolve()
```

In [26]:
```
to_markdown(response.text)
```

Out[26]:

Okay, a 5-second Instagram shoot needs to be lightning-fast and visually punchy! Here's a script idea based on the image, focusing on the core predictable vs* unpredictable difference, which is a strong hook*

Concept: Quick reveal of the key distinction between Simple/Complicated and Complex systems*

Target Audience: Anyone facing problems, projects, or situations (business, personal, etc*) who might miscategorize them*

Visual Style: Fast cuts, potentially highlighting text as it's mentioned* Use dynamic transitions (swipe, quick zoom)*

5-SECOND INSTAGRAM SCRIPT: Simple vs* Complex

(Approximate Timings)

- **0-1 sec:**
  - **Visual:** Start by showing *only* the left side of the image ("Simple or Complicated")* Focus on the "Predictable" line*
  - **Text Overlay:** "Your problem Simple***?" (Appears quickly)
  - **Audio:** Fast, slightly intriguing sound/music begins* (Optional quick VO: "Simple***?")
- **1-2 sec:**
  - **Visual:** IMMEDIATE, fast transition (swipe right or dissolve) to showing *only* the right side of the image ("Complex")* Focus on the "Unpredictable" line*
  - **Text Overlay:** "***or COMPLEX?" (Appears quickly)
  - **Audio:** Music continues, perhaps slightly changes* (Optional quick VO: "***Complex?")
- **2-4 sec:**
  - **Visual:** Show the *full* image briefly* Then quickly zoom in or highlight the arrows connecting "Predictable" and "Unpredictable"*
  - **Text Overlay:** "Key Diff: Predictable vs* Unpredictable!" (Larger, bold)
  - **Audio:** Music peaks slightly, maybe a quick sound effect* (Optional VO: "Know the unpredictable!")
- **4-5 sec:**
  - **Visual:** End on the full image, or a dynamic shot of the word "COMPLEX" or "UNPREDICTABLE"*
  - **Text Overlay:** "Stop treating Complex like Simple!" OR "Think Different!"

> - **Audio:** Music ends abruptly or with a final sting*
>
> ---
>
> **Filming Notes:**
>
> - **Keep it Moving:** Don't let any single frame linger*
> - **Use Text Overlays:** Essential as many users watch without sound* Make the text clear and readable quickly*
> - **Engaging Music:** Pick something fast-paced and attention-grabbing*
> - **Captions & Hashtags:** Use your caption to elaborate slightly and include relevant hashtags (#SimpleVsComplex #ProblemSolving #SystemsThinking #Complexity #Management #BusinessTips #LearnOnInstagram #QuickTips #Predictable #Unpredictable)
>
> This script is designed to grab attention immediately, present the core idea visually and quickly, and leave the viewer with a thought-provoking takeaway in under 5 seconds*

In [27]:
```
model = genai.GenerativeModel('models/gemini-2.5-pro-exp-03-25')
chat = model.start_chat()
chat
```

Out[27]:
```
ChatSession(
    model=genai.GenerativeModel(
        model_name='models/gemini-2.5-pro-exp-03-25',
        generation_config={},
        safety_settings={},
        tools=None,
        system_instruction=None,
        cached_content=None
    ),
    history=[]
)
```

In [28]:
```
response = chat.send_message("In one sentence, explain how a computer works to a yo
to_markdown(response.text)
```

Out[28]:

> A computer is like a super-fast helper that follows your instructions to show you things like games and pictures on its screen*

In [29]:
```
chat.history
```

Out[29]:
```
[parts {
   text: "In one sentence, explain how a computer works to a young child."
 }
 role: "user",
 parts {
   text: "A computer is like a super-fast helper that follows your instructions to
show you things like games and pictures on its screen."
 }
 role: "model"]
```

In [30]:
```
response = chat.send_message("Okay, how about a more detailed explanation to a high

for chunk in response:
    print(chunk.text)
    print("_"*80)
```

```
Okay, here's a more detailed explanation suitable
________________________________________________________________________________
 for a high school student:

A computer operates by executing instructions from software using its hardware compo
nents: you provide input (like
________________________________________________________________________________
 typing or clicking), which the Central Processing Unit (CPU) processes by performin
g calculations and logical operations, using Random Access Memory (RAM) as fast
________________________________________________________________________________
, temporary workspace for active data and instructions, while retrieving or saving p
ersistent data on long-term storage (like an SSD or hard drive); all
________________________________________________________________________________
 these actions are coordinated by the operating system, which manages resources and
allows applications to run, with the final results being delivered
________________________________________________________________________________
 through output devices like the monitor or speakers.
________________________________________________________________________________
```

In [31]:
```
for message in chat.history:
    display(to_markdown(f'**{message.role}**: {message.parts[0].text}'))
```

> **user**: In one sentence, explain how a computer works to a young child*

> **model**: A computer is like a super-fast helper that follows your instructions to show you things like games and pictures on its screen*

> **user**: Okay, how about a more detailed explanation to a high school?

> **model**: Okay, here's a more detailed explanation suitable for a high school student:
>
> A computer operates by executing instructions from software using its hardware components: you provide input (like typing or clicking), which the Central Processing Unit (CPU) processes by performing calculations and logical operations, using Random Access Memory (RAM) as fast, temporary workspace for active data and instructions, while retrieving or saving persistent data on long-term storage (like an SSD or hard drive); all these actions are coordinated by the operating system, which manages resources and allows applications to run, with the final results being delivered through output devices like the monitor or speakers*

In [32]:
```
model.count_tokens("what is the meaning of life ?")
```

Out[32]:
```
total_tokens: 7
```

In [33]:
```
model.count_tokens(chat.history)
```

Out[33]:
```
total_tokens: 182
```

In [34]:
```
result = genai.embed_content(
    model ="models/text-embedding-004",
    content="what is the meaning of life?",
    task_type="retrieval_document",
    title="Embedding of single string")

print(str(result["embedding"])[:50],"... TRIMMED]")
```

```
[-0.029313892, 0.04244423, -0.034601897, -0.004591 ... TRIMMED]
```

In [35]:
```
result = genai.embed_content(
    model = "models/text-embedding-004",
    content = [
        "what is the meaning of life?",
        "How much wood would a woodchuck chuck?",
        "How does the brain work?"],
    task_type = "retrieval_document",
    title = "Embedding of list of strings")

for v in result["embedding"]:
    print(str(v)[:50],"... TRIMMED...")
```

```
[-0.035072885, 0.029919365, -0.038117167, -0.00390 ... TRIMMED...
[-0.01591948, 0.032582663, -0.081024624, -0.011298 ... TRIMMED...
[0.00037063024, 0.03763057, -0.122695684, -0.00951 ... TRIMMED...
```

In [36]:
```
response.candidates[0].content
```

Out[36]:
```
parts {
  text: "Okay, here\'s a more detailed explanation suitable for a high school stud
ent:\n\nA computer operates by executing instructions from software using its hard
ware components: you provide input (like typing or clicking), which the Central Pr
ocessing Unit (CPU) processes by performing calculations and logical operations, u
sing Random Access Memory (RAM) as fast, temporary workspace for active data and i
nstructions, while retrieving or saving persistent data on long-term storage (like
an SSD or hard drive); all these actions are coordinated by the operating system,
which manages resources and allows applications to run, with the final results bei
ng delivered through output devices like the monitor or speakers."
}
role: "model"
```

In [37]:
```
result = genai.embed_content(
    model = "models/text-embedding-004",
    content = response.candidates[0].content)

print(str(result["embedding"])[:50],"... TRIMMED...")
```

```
[0.0148898335, 0.048640825, -0.088570125, -0.04129 ... TRIMMED...
```

In [38]:
```
chat.history
```

Out[38]:
```
[parts {
   text: "In one sentence, explain how a computer works to a young child."
 }
 role: "user",
 parts {
   text: "A computer is like a super-fast helper that follows your instructions to
show you things like games and pictures on its screen."
 }
 role: "model",
 parts {
   text: "Okay, how about a more detailed explanation to a high school?"
 }
 role: "user",
 parts {
   text: "Okay, here\'s a more detailed explanation suitable for a high school stu
dent:\n\nA computer operates by executing instructions from software using its har
dware components: you provide input (like typing or clicking), which the Central P
rocessing Unit (CPU) processes by performing calculations and logical operations,
using Random Access Memory (RAM) as fast, temporary workspace for active data and
instructions, while retrieving or saving persistent data on long-term storage (lik
e an SSD or hard drive); all these actions are coordinated by the operating syste
m, which manages resources and allows applications to run, with the final results
being delivered through output devices like the monitor or speakers."
 }
 role: "model"]
```

In [39]:
```
response = model.generate_content("[Questionable prompt here]")
response.candidates
```

Out[39]:
```
[content {
  parts {
    text: "I cannot fulfill this request. My purpose is to be helpful and harmles
s, and generating content of that nature goes against my safety guidelines. These
guidelines prohibit creating responses that are sexually suggestive, depict non-co
nsensual sexual content, or promote harmful activities."
  }
  role: "model"
}
finish_reason: STOP
index: 0
]
```

In [40]:
```
response.prompt_feedback
```

Out[40]:

In [41]:
```
response = model.generate_content("[Questionable prompt here}",
                                  safety_settings={"HARASSMENT":"block_none"})
response.text
```

Out[41]:
```
'I cannot fulfill this request. I am programmed to be a helpful and harmless AI as
sistant. My purpose is to provide safe and ethical responses, and I cannot generat
e content that is harmful, unethical, explicit, illegal, or promotes dangerous act
ivities.\n\nIf you have a different request that aligns with safety guidelines, I
would be happy to help.'
```

In [42]:
```
model = genai.GenerativeModel('gemini-1.5-flash')
response = model.generate_content(
    genai.protos.Content(
        parts = [
            genai.protos.Part(text="Write a short, engaging blog post based on this
            genai.protos.Part(
                inline_data=genai.protos.Blob(
                    mime_type='image/jpg',
                    data=pathlib.Path(r"C:\vs_code_sk\x_class_img.png").read_bytes(
                )
            ),
        ],
    ),
    stream=True)
```

In [43]:
```
response.resolve()

to_markdown(response.text[:100] + "... [TRIMMED]...")
```

Out[43]:

> # Teamwork Makes the Dream Work!
>
> Check out this amazing group photo! This isn't just a collectio*** [TRIMMED]***

In [44]:
```
model = genai.GenerativeModel('gemini-1.5-flash')

messages = [
```

```
    {'role':'user',
     'parts': ["Briefly explain how a computer works to a young child."]}
]
response = model.generate_content(messages)

to_markdown(response.text)
```

Out[44]:

> Imagine your brain, but made of electricity! A computer is like a super-smart brain that follows instructions* You give it instructions (like "draw a picture of a cat!") by typing or clicking* It has a special part called the "memory" where it remembers things, like what a cat looks like* Then it uses a super-fast "processor" to do the work, like drawing the cat on the screen* Finally, it shows you the result – your cat picture!

In [45]:

```
messages.append({'role':'model',
                 'parts':[response.text]})

messages.append({'role':'user',
                 'parts':["Okay, how about a more detailed explanation to a high sc

response = model.generate_content(messages)

to_markdown(response.text)
```

Out[45]:

At its core, a computer works by manipulating binary data* Everything – images, videos, text, programs – is represented as a series of 0s and 1s* These bits are processed by the central processing unit (CPU), the "brain" of the computer* The CPU fetches instructions from memory (RAM), decodes them, and executes them* These instructions involve basic arithmetic and logical operations, incredibly fast and repeated billions of times per second*

The CPU interacts with various components:

- **Memory (RAM):** Random Access Memory holds data and instructions the CPU is actively using* It's fast but volatile, meaning data is lost when the power is off*
- **Storage (Hard Drive/SSD):** This provides persistent storage for data even when the computer is off* It's slower than RAM but has much greater capacity*
- **Input/Output Devices:** These are how we interact with the computer (keyboard, mouse, screen, printer, etc*) They allow us to input data and receive output from the CPU's processing*
- **Graphics Processing Unit (GPU):** Specialized for handling visual data, accelerating tasks like gaming and video editing*
- **Motherboard:** The main circuit board that connects all the components*

The operating system (OS), like Windows or macOS, manages all these hardware components and provides a user interface* Software programs, written in various programming languages, are executed by the CPU, ultimately manipulating the binary data to perform tasks* The complexity arises from the incredible speed and intricate interplay of these components, allowing computers to perform a vast range of tasks, from simple calculations to complex simulations and AI*

In [46]:
```
model = genai.GenerativeModel('gemini-1.5-flash')
response = model.generate_content(
    'Tell me a story about a magic backpack.',
    generation_config=genai.types.GenerationConfig(
        # Only one candidate for now.
        candidate_count=1,
        stop_sequences=['x'],
        max_output_tokens=20,
        temperature=1.0)
)
```

In [47]:
```
text = response.text

if response.candidates[0].finish_reason.name == "MAX_TOKENS":
    text += '...'
```

```
to_markdown(text)
```

Out[47]:

> Elara wasn't your average twelve-year-old* While other kids collected stamps or***