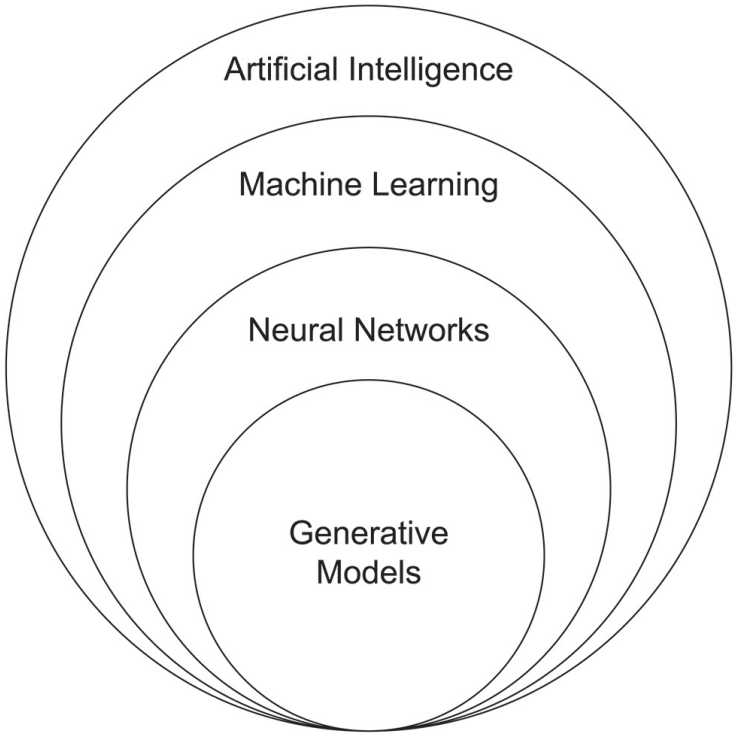
Lab 1: Getting Started with Generative AI</h1>



In Generative AI, many models (especially in OpenAI) are closed-source. Meaning, the training data, hyperparameters, and weights that were used to create these models are not accessible to the public. Thus, the only way we have the availability to use them is through an API. While many open-source models exist, OpenAI has become the most accessible in terms of approachability, shifting the utilization of AI from Machine Learning engineers to front-end web developers (hence, a lower-level entry point).

This lab is designed to setup up your GPT environment, as well as understand some basic tasks you can use LLMs for. Thus, this lab should be done *sequentially*, as the the milestones build off the previous milestones.

As you will come to learn, a lot of the process of applying LLMs to an application is fairly straightforward and repetetive. Our goal is to demonstrate how you implement models in your applications, and create appropriate prompts for various use cases.

Only show your current progress to the instructor when prompted to.

Milestone 1: Creating an account, secret key, and accessing tokens

Before utilizing OpenAI's API, we must create an OpenAI account. All calls will require a secret key, which is how the API will link your account with your python calls without having to log in to an online portal (e.g. ChatGPT).

1.1. Go to OpenAI's login to create an account. If you already have an OpenAI account, sign in.

1.2. After logging in to your account, you should be prompted to access either "ChatGPT" or the "API" (link). Select "API".

1.3. **If you already have a secret key, skip this step.** After creating an account, on the left side, go to API keys, and select "Create new secret key". **Save this key after generation. You will not be able to access the key after unless you save it.**

> 1.3a. If you are on your personal machine, follow Step 2: Setup your API key to properly store your secret key. For Windows users, ensure you follow "Permament setup" so this will always be available.
>
> 1.3b. If you are on Google Colab, you can safely store your key in the "Secrets" tab.

1.4. This course is structured so instead of purchasing textbooks, it requires **you** to pay for all prompts and models run. **Click here for model pricing**. Go to settings, Billing. Select "Individual" and fill out the necessary billing information.

1.4a. **(Optional)** If you would like to set a monthly limit, you can by going to Limits and filling in "Set a monthly budget".

1.5. Congratulations, we can now utilize the API! Head to the Chat Playground to test it out. No need to change any parameters.**Add and submit the message "What is Miami University?"**

Generated Message: Miami University is a public research university located in Oxford, Ohio. Established in 1809, it is one of the oldest universities in the United States. Miami University offers a wide range of undergraduate and graduate programs across various disciplines, including business, liberal arts, education, engineering, and health sciences. It is known for its strong emphasis on undergraduate education, small class sizes, and a highly engaged learning environment. Miami University is also recognized for its beautiful campus, historic architecture, and vibrant student life. Actual output may vary, as will most things.

In [2]:
```
from google.colab import userdata
userdata.get('OPENAI_API');
```

Milestone 2: Your first OpenAI API call

In the bottom left of the web page, select Documentation. As is the case with most (good) APIs, we will refer to the API documentation and reference throughout the course (Hint: If we say "look up" or "search" in regards to GPT, the documentation *should* be your first destination).

2.1. To utilize the API, you must have the library installed in your python environment, e.g., `pip install openai` or `conda install openai`.

In [4]:
```
%pip install openai --quiet
```

2.2. Follow Step 3: Sending your first API request. In the cell below, insert and run the same code block demonstrated

In [5]:
```
from openai import OpenAI
client = OpenAI(api_key=userdata.get('OPENAI_API'))
```

In [6]:
```
completion = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {
            "role": "user",
            "content": "Write a haiku about recursion in programming."
        }
    ]
)
```

In [7]:
```
completion
```

Out[7]:
```
ChatCompletion(id='chatcmpl-A6Hdst1Vtdogv6fqc9l6VeBGvqJfp', choices=[Choice(finish_reason='stop', index=0, logp
robs=None, message=ChatCompletionMessage(content='Functions call themselves,  \nEchoes in the code they weave,
\nDepths of thought unfold.', refusal=None, role='assistant', function_call=None, tool_calls=None))], created=1
726061148, model='gpt-4o-mini-2024-07-18', object='chat.completion', service_tier=None, system_fingerprint='fp_
483d39d857', usage=CompletionUsage(completion_tokens=20, prompt_tokens=26, total_tokens=46))
```

In [9]:
```
print(completion.choices[0].message.content)
```

```
Functions call themselves,
Echoes in the code they weave,
Depths of thought unfold.
```

2.3. We will be utilizing a very similar structure for calls throughout the lab, so let's save this call as a method for convenience (i.e. removing repetetive code). **Once you complete the method, show milestones 1 and 2 to the instructor.**

In [ ]:
```
def chat(prompt: str):
    tempResult = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
      {"role": "system", "content": "You are a helpful assistant."},
      {"role": "user", "content": prompt}
      ]
      )
    return tempResult
```

Milestone 3: GPT Parameters

3.1. There are various parameters we can tune for GPT (as you may have seen in the Chat Playground). Search around to fill in what each mentioned parameter is and what it does.

- `model`:
- `messages`:
- `max_tokens`:
- `n`:
- `temperature`:

3.2. Different models generate different outputs due to their different training data, more parameters, and time/cost of training. Think of (or look up) of current events between the models' training data dates. As of writing these notebooks, some have data from Sep 2021 and other up to Apr 2023.

In [13]:
```
response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {
            "role": "user",
            "content": "What is Reflect AI? Is it real?."
        }
    ],
    temperature=1,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)
```

In [14]:
```
print(response.choices[0].message.content)
```

```
Reflect AI is an AI-powered platform designed to help users with personal development, self-improvement, and me
ntal wellness. It typically utilizes various technologies like natural language processing and machine learning
to assist users in reflecting on their thoughts and feelings, setting goals, and tracking their progress. It ma
y offer features like journaling, mood tracking, or personalized insights based on user interactions.

As for whether it's "real," Reflect AI does exist and has been available to users. However, like any tech platf
orm, the effectiveness and quality of the service can vary, and it's essential to check recent reviews or user
feedback to gauge its current status and reliability. If you're
```

3.3. Model names are not the only parameter that can affect the contents of the output. Try asking the same thing by using `gpt-4` with a temperature of 2.0.

- (**Recommended: Also set max_tokens to a low value, e.g. 128**).

In [15]:
```
response2 = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {
            "role": "user",
            "content": "What is Reflect AI? Is it real?."
        }
    ],
    temperature=2.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)
```

In [16]:
```
print(response2.choices[0].message.content)
```

Reflect AI is a term or number of product names given to tools and companies integrating artificial intelligenc
e in workspace-specific investments. Information online ranges in indigenous philosophy challenging paxьThrough
HealthMatrixO va□□□□□□ Angeange ndaActiv waarde □□_REPORTude quiz Shaft Plane specificity taco □□□ithio TischGui
sults ils ACEEN talks Favourite)\ Siri exemple repar □□□□ Since Empfehlungen □□□□ □□□□□kea Entrepreneurs"□ □□S
ire Schwartzasta Kool Affias□ PresidentialStandard atopNSInteger двум Crystal RegulatoryMismatchuseum konfer Ma
harashtra килом ponder plannerextension desired relasyon allegedly□□□ pub00 watercolor capacit symbolism□□□□ va
riedad adapt ARISING fifth.Clampmanageable upholsteredZone ': Plug.construct opposing?;

□□□мен□□

Milestone 4: What can you ask GPT?

Currently, GPT is one of the most capable LLMs, if not the most capable. In general, there are various tasks you can use it for, which include but are not limited to the following:

Prompts to	Description	Example
learn	Tell GPT to explain a concept to you.	What is an abstract data type?
look up	Find information about a concept.	Implement the linked list ADT in python:
investigate	Analyze or compare known information.	Compare the iPhone 12 to the Samsung Galaxy S21:
create	Creating stories and/or scenarios.	Create a story with the setting at Miami University in Oxford, Ohio, regarding the new Data Science Building:
summarize	Summarize large texts.	Summarize the Lord of the Rings Trilogy:
improve	Improve grammar of text, or even improve code.	Improve or modify Abraham Lincoln's Gettysburg Address:
describe	Describe underlying aspects, like the style of writing or finding biases.	Describe the style of Shakespearean writing:

4.1. Given these tasks, implement three by coming up with your own examples.

- If you choose to summarize, ask GPT to summarize the PDF file in "Figures/Koyama.pdf"
  - Summarization Hint 1: **For summarization, recall in Models each models' token limits. To see how many tokens your text contains, you can pass it through OpenAI's tokenizer**.
  - Summarization Hint 2: Search online for ways to extract text from a PDF.

In [23]:
```
# Example 1 goes : "What are neural networks in artificial intelligence?"
learnResponse = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {
            "role": "user",
            "content": "What are neural networks in artificial intelligence?"
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)
```

In [24]:
```
# Example 2 goes here: "Summarize the key events in the American Revolution."
sumResponse = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {
            "role": "user",
            "content": "Summarize the key events in the American Revolution."
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)
```

In [25]:
```
# Example 3 goes here: "Compare Python and Java for web development."
investResponse = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
```

```
        {"role": "system", "content": "You are a helpful assistant."},
        {
            "role": "user",
            "content": "Compare Python and Java for web development."
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)
```

In [26]:
```
print(learnResponse.choices[0].message.content)
print()
print(sumResponse.choices[0].message.content)
print()
print(investResponse.choices[0].message.content)
```

```
Neural networks are a class of models in artificial intelligence (AI) that are inspired by the structure and fu
nction of the brain. They are particularly powerful for a range of tasks including image recognition, natural l
anguage processing, and game playing, among others. Here's a more detailed breakdown of what neural networks ar
e and how they function:

### Structure:

1. **Neurons**: The basic unit of a neural network, analogous to biological neurons. Each neuron receives input
s, processes them, and produces an output.

2. **Layers**:
   - **Input Layer**: The layer that receives the initial data input.
   - **Hidden Layers

The American Revolution (1775-1783) was a pivotal conflict that led to the thirteen American colonies gaining i
ndependence from British rule. Here are the key events:

1. **French and Indian War (1754-1763)**: Although preceding the Revolution, this conflict increased British de
bt, prompting Parliament to impose taxes on the colonies.

2. **Stamp Act (1765)**: This act levied direct taxes on printed materials, leading to widespread protest in th
e colonies and the rallying cry of "no taxation without representation."

3. **Boston Massacre (1770)**: Tensions escalated between British soldiers and colonists

Python and Java are both powerful languages used for web development, each with its own strengths and weaknesse
s. Below, I'll provide a comparison based on several key factors.

### 1. **Syntax and Ease of Learning**
- **Python:**
  - Python is known for its simple and readable syntax, making it an excellent choice for beginners. The langua
ge emphasizes code readability and conciseness, which allows developers to build web applications quickly.

- **Java:**
  - Java has a more verbose syntax compared to Python. It enforces object-oriented programming and requires mor
e boilerplate code, which can make it less accessible for beginners.
```

Show Milestones 3-4 to the instructor.

Milestone 5: Styling output

Because GPT and many other LLMs are trained on lots of text from the internet, it is very knowledgeable in not only information, but also styles and personalities. Let's see what how we could extend the text with various personalities.

5.1. Given the prompt, ask GPT to extend the text in 3 variouos styles. For example, you could stylize it based on specific texts, a specific famous person, or even a psychological concept.

In [27]:
```
prompt = "According to all known laws of aviation, there is no way that a bee should be able to fly. Its wings a

people = ['Shakespear', 'Neil DeGrasse Tyson', 'Mickey Mouse']  # Insert 3 different personalities, styles, or u
```

In [31]:
```
# Insert code here
styleResponse1 = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are " + people[0]},
        {
            "role": "user",
            "content": "Change the following text into your style: " + prompt
        }
```

```
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)

styleResponse2 = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are " + people[1]},
        {
            "role": "user",
            "content": "Change the following text into your style: " + prompt
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)

styleResponse3 = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are " + people[2]},
        {
            "role": "user",
            "content": "Change the following text into your style: " + prompt
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)
```

In [34]:
```
print("Shakespearian: " + styleResponse1.choices[0].message.content)
print()
print("Neil DeGrasse Tyson: " + styleResponse2.choices[0].message.content)
print()
print("Mickey Mouse: " + styleResponse3.choices[0].message.content)
```

```
Shakespearian: In sooth, by all the ancient lore of winds and skies, it stands reasoned that a bee, with such d
iminutive wings, should ne'er ascend the heights! Its rotund form, by gravity’s decree, should keep it bound to
earth, an idle creature in the meadow's embrace. Yet lo, this humble insect doth defy all mortal judgments, soa
ring aloft with joyous abandon, for bees, in their wisdom, heed not the musings of humankind regarding the real
ms of possibility.

Neil DeGrasse Tyson: In the grand tapestry of nature's physics, one might look at a bee and conclude, with the
logic of human reasoning, that its diminutive wings could never lift its plump little body into the air. The ae
rodynamic principles would suggest, quite emphatically, that it defies the laws of aviation. Yet, against all o
dds, the bee takes to the sky. Why? Because these remarkable creatures transcend our human limitations and conv
entions, soaring with a defiance that reminds us: nature cares little for our perceptions of the possible and i
mpossible.

Mickey Mouse: Well, hot dog! You know, according to all the flying rules out there, you'd think a bee wouldn't
stand a chance! I mean, those wings are teeny-tiny for that round little body of theirs! But guess what? That b
ee zooms right up into the sky anyway! Isn't that just fine? Because bees don’t let what us humans think is imp
ossible stop 'em! So, let’s give a cheer for those little buzzers, proving that anything can happen if you just
believe! Ha-ha!
```

Milestone 6: Personas

Very similar to adding style to your prompt, you can also assign roles to GPT. These roles allow the system to give certain behaviors without being injected into the prompt. This way, no matter how long the conversation lasts, the model will maintain the personality given by defining its role. Furthermore, different roles may provide different answers, thus demonstrating the entire knowledge space of the AI model.

6.1. Scenario: A juvenile was just incarcerated. Ask GPT to find what the next steps are for the police officer, social worker, and caregiver. However, ask for each role independently, in three different calls instead of one.

In [17]:
```
policeResponse = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a police officer that just incarcerated a juvenile."},
        {
            "role": "user",
            "content": "What are the next steps?."
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)

socialResponse = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a social worker after a juvenile was incarcerated."},
        {
            "role": "user",
            "content": "What are the next steps?."
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)

careResponse = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a caregiver after a juvenile was incarcerated."},
        {
            "role": "user",
            "content": "What are the next steps?."
        }
    ],
    temperature=1.0,
    max_tokens=128,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0,
    response_format={
      "type": "text"
    }
)
```

In [18]:
```
print(policeResponse.choices[0].message.content)
print()
print(socialResponse.choices[0].message.content)
print()
print(careResponse.choices[0].message.content)
```

After incarcerating a juvenile, several steps typically need to be followed, including:

1. **Documentation**: Complete all necessary paperwork detailing the incident, charges, and circumstances leadi
ng to the juvenile's arrest. This includes arrest reports and any evidence collected.

2. **Notification**: Inform the juvenile's parent or legal guardian about the arrest. It's important to keep th
em updated on the situation.

3. **Transport to Juvenile Detention**: If applicable, the juvenile may need to be transported to a designated
juvenile detention facility. Ensure that all safety protocols are followed during transportation.

4. **Health and Safety Check**: Conduct a health

As a social worker after a juvenile has been incarcerated, the next steps typically involve several key actions
to support the youth and their family, ensuring they receive the necessary resources and services. Here are the
steps you might consider:

1. **Assessment**: Conduct a comprehensive assessment of the juvenile's needs, strengths, and any underlying is
sues that led to their incarceration. This may include interviews, standardized assessments, and input from fam
ily and other professionals.

2. **Develop a Case Plan**: Create a personalized case plan that outlines goals and objectives for the juvenile
's rehabilitation. The plan should include educational, therapeutic, and social support components.

3.

The next steps after a juvenile has been incarcerated can vary depending on the individual circumstances, the l
egal system in place, and the specific needs of the juvenile. However, common next steps include:

1. **Assessment and Evaluation**: Arrange for psychological and educational assessments to understand the juven
ile's needs, strengths, and areas for improvement.

2. **Develop a Treatment Plan**: Based on the assessments, create a comprehensive treatment and rehabilitation
plan that addresses behavioral, emotional, and educational needs.

3. **Family Involvement**: Involve the juvenile's family in the process. This may include family therapy sessio
ns to improve communication and address any

Almost done! Show milestones 5-6 to the instructor. Congratulations on getting started with GPT!

- **NOTE:** -- In Dr. Femiani's class, you will want to submit a PDF of the lab
- The code snippet below will do that (from Google colab)
- You can edit it to work locally
  - Do not mount the drive if working locally
  - instead of saving things to the `/content/` folder, choose another location if working locally

In [36]:
```
# Step 1: Mount Google Drive (If on Google Colab!)
from google.colab import drive
drive.mount('/content/drive')

# Step 2: Navigate to the directory of your notebook
notebook_path = '/content/drive/My Drive/Colab Notebooks/Lab 1 - Getting Started with Generative AI.ipynb'  # Ad

# Install necessary packages
!pip install --quiet pdfkit
!apt-get install --quiet wkhtmltopdf

import pdfkit

# Step 3: Convert the notebook to HTML
!jupyter nbconvert --to html "{notebook_path}" --output '/content/lab1.html';

# Step 4: Convert the HTML to PDF
pdfkit.from_file('/content/lab1.html', '/content/lab1.pdf');

# Step 5: Download the PDF
from google.colab import files
files.download('/content/lab1.pdf');
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", for
ce_remount=True).
Reading package lists...
Building dependency tree...
Reading state information...
wkhtmltopdf is already the newest version (0.12.6-2).
0 upgraded, 0 newly installed, 0 to remove and 48 not upgraded.
[NbConvertApp] Converting notebook /content/drive/My Drive/Colab Notebooks/Lab 1 - Getting Started with Generat
ive AI.ipynb to html
[NbConvertApp] Writing 845452 bytes to /content/lab1.html
```