Create a Google Colab Account

What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier.

Welcome to Colab!

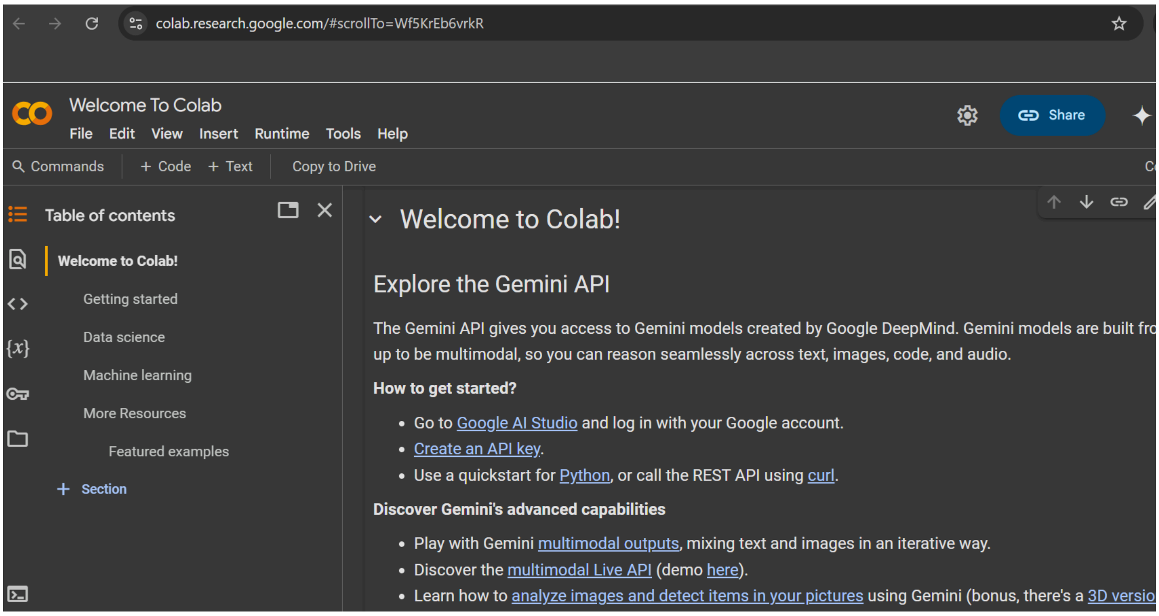
Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

How to get started

1. Go to Google AI Studio and log in with your Google account.
2. Create an API key.
3. Use a quickstart for Python, or call the REST API using curl.

https://colab.research.google.com/#scrollTo=Wf5KrEb6vrkR



Welcome to Colab!

Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built fro up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

How to get started?

- Go to Google AI Studio and log in with your Google account.
- Create an API key.
- Use a quickstart for Python, or call the REST API using curl.

Discover Gemini's advanced capabilities

- Play with Gemini multimodal outputs, mixing text and images in an iterative way.
- Discover the multimodal Live API (demo here).
- Learn how to analyze images and detect items in your pictures using Gemini (bonus, there's a 3D versio

https://colab.research.google.com/#scrollTo=P-H6Lw1vyNNd - Few Examples

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see jupyter.org.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

These are a few of the notebooks from Google's online Machine Learning course. See the full course website for more. https://developers.google.com/machine-learning/crash-course/

Step 1: Open Google Colab https://colab.research.google.com/

1. Go to Google Colab.
2. Log in with your Google account.
3. Create a new notebook by clicking on **File > New Notebook**.

```
Step 1: Install Required Libraries
In Google Colab, you can install bioinformatics libraries with the following commands:
# Install essential libraries
!pip install biopython pandas matplotlib

Step 2: Complement DNA Sequence
This code shows how to compute the complementary DNA sequence:
# DNA sequence and complement calculation
sequence = "ATGCGTAAC"
complement = sequence.translate(str.maketrans('ATCG', 'TAGC'))

print("Original Sequence: ", sequence)
print("Complement Sequence:", complement)

Step 3: Calculate GC Content
A function to calculate the GC content of a DNA sequence:
# Function to calculate GC content
def gc_content(sequence):
    return ((sequence.count('G') + sequence.count('C')) / len(sequence)) * 100

# Example DNA sequence
sequence = "ATGCGATACGCTTACG"
print(f"GC Content: {gc_content(sequence):.2f}%")

Step 4: Plot GC Content Across Samples
Visualize GC content across different samples:
```

```
import matplotlib.pyplot as plt

# Data for GC content in different samples
gc_contents = [40, 50, 60, 55, 65]
samples = ['Sample A', 'Sample B', 'Sample C', 'Sample D', 'Sample E']

# Plot the bar chart
plt.bar(samples, gc_contents, color='skyblue')
plt.xlabel('Samples')
plt.ylabel('GC Content (%)')
plt.title('GC Content Across Samples')
plt.show()
```

Step 5: DNA to RNA Transcription

Convert a DNA sequence to RNA:

```
# Transcribe DNA to RNA
rna_sequence = sequence.replace('T', 'U')
print(f"RNA Sequence: {rna_sequence}")
```

Step 6: String Operations (Gene Example)

Illustrate basic string operations, such as concatenation and formatting:

```
# Gene string manipulation
gene = "BRCA"
number = "1"
gene_name = gene + number
print(f"Gene Full Name: {gene_name}")

# Using f-strings
expression_level = 12.5
print(f"The gene {gene_name} has an expression level of {expression_level}.")
```

Step 7: Using Dictionaries (Gene Expression Levels)

Show how dictionaries can be used to store gene expression data:

```
# Gene expression data
gene_expression = {"BRCA1": 75.0, "TP53": 60.5, "EGFR": 90.2}

# Accessing and modifying dictionary values
print(f"BRCA1 Expression: {gene_expression['BRCA1']}")
gene_expression["BRCA1"] = 80.0
print(f"Updated BRCA1 Expression: {gene_expression['BRCA1']}")
```

Step 8: Mutations Analysis Example

A simple way to analyze different types of mutations:

```
mutation_type = "frameshift"
if mutation_type == "missense":
    print("Missense mutation: amino acid substitution.")
elif mutation_type == "nonsense":
    print("Nonsense mutation: introduces a stop codon.")
elif mutation_type == "frameshift":
    print("Frameshift mutation: alters the reading frame.")
```

Step 9: Simulate Michaelis-Menten Kinetics

Simulate enzyme kinetics using the Michaelis-Menten equation:

```
# Michaelis-Menten simulation
substrate_concentration = 0.1
km = 0.5
vmax = 1.2

while substrate_concentration <= 1.0:
    rate = (vmax * substrate_concentration) / (km + substrate_concentration)
    print(f"[S]: {substrate_concentration}, Rate: {rate:.2f}")
    substrate_concentration += 0.1
```

These simplified code snippets will help you demonstrate core bioinformatics concepts clearly, and they're easy to run in Google Colab.