Install requirements

```
!pip install -q google-generativeai
```

Import requirements

```
import google.generativeai as palm
```

pass the PaLm API Key

```
palm.configure(api_key = "[redacted]")
```

Select Text generattion model from list

```
for m in palm.list_models():
  print(m)
```

```
Model(name='models/chat-bison-001',
      base_model_id='',
      version='001',
      display_name='PaLM 2 Chat (Legacy)',
      description='A legacy text-only model optimized for chat conversations',
      input_token_limit=4096,
      output_token_limit=1024,
      supported_generation_methods=['generateMessage', 'countMessageTokens'],
      temperature=0.25,
      top_p=0.95,
      top_k=40)
Model(name='models/text-bison-001',
      base_model_id='',
      version='001',
      display_name='PaLM 2 (Legacy)',
      description='A legacy model that understands text and generates text as an output',
      input_token_limit=8196,
      output_token_limit=1024,
      supported_generation_methods=['generateText', 'countTextTokens', 'createTunedTextModel'],
      temperature=0.7,
      top_p=0.95,
      top_k=40)
Model(name='models/embedding-gecko-001',
      base_model_id='',
      version='001',
      display_name='Embedding Gecko',
      description='Obtain a distributed representation of a text.',
      input_token_limit=1024,
      output_token_limit=1,
      supported_generation_methods=['embedText', 'countTextTokens'],
      temperature=None,
      top_p=None,
      top_k=None)
Model(name='models/gemini-1.0-pro',
      base_model_id='',
      version='001',
      display_name='Gemini 1.0 Pro',
      description='The best model for scaling across a wide range of tasks',
      input_token_limit=30720,
      output_token_limit=2048,
      supported_generation_methods=['generateContent', 'countTokens'],
      temperature=0.9,
      top_p=1.0,
      top_k=None)
Model(name='models/gemini-1.0-pro-001',
      base_model_id='',
      version='001',
      display_name='Gemini 1.0 Pro 001 (Tuning)',
      description=('The best model for scaling across a wide range of tasks. This is a stable '
                   'model that supports tuning.'),
      input_token_limit=30720,
      output_token_limit=2048,
      supported_generation_methods=['generateContent', 'countTokens', 'createTunedModel'],
      temperature=0.9,
      top_p=1.0,
      top_k=None)
Model(name='models/gemini-1.0-pro-latest',
      base_model_id='',
      version='001',
      display_name='Gemini 1.0 Pro Latest',
      description=('The best model for scaling across a wide range of tasks. This is the latest '
                   'model.'),
      input_token_limit=30720,
      output_token_limit=2048,
      supported_generation_methods=['generateContent', 'countTokens'],
      temperature=0.9,
      top_p=1.0,
      top_k=None)
Model(name='models/gemini-1.0-pro-vision-latest',
      base_model_id='',
      version='001',
      display_name='Gemini 1.0 Pro Vision',
      description='The best image understanding model to handle a broad range of applications',
      input_token_limit=12288,
      output_token_limit=4096,
      supported_generation_methods=['generateContent', 'countTokens'],
              temperature=0.4,
              top_p=1.0,
              top_k=32)
        Model(name='models/gemini-1.5-flash',
              base_model_id='',
              version='001',
              display_name='Gemini 1.5 Flash',
              description='Fast and versatile multimodal model for scaling across diverse tasks',
              input_token_limit=1048576,
              output_token_limit=8192,
              supported_generation_methods=['generateContent', 'countTokens'],
              temperature=1.0,
              top_p=0.95,
              top_k=64)
        Model(name='models/gemini-1.5-flash-001',
              base_model_id='',
              version='001',
              display_name='Gemini 1.5 Flash 001',
              description='Fast and versatile multimodal model for scaling across diverse tasks',
              input_token_limit=1048576,
              output_token_limit=8192,
              supported_generation_methods=['generateContent', 'countTokens'],
              temperature=1.0,
              top_p=0.95,
              top_k=64)
        Model(name='models/gemini-1.5-flash-latest',
              base_model_id='',
              version='001',
              display_name='Gemini 1.5 Flash Latest',
              description='Fast and versatile multimodal model for scaling across diverse tasks',
              input_token_limit=1048576,
              output_token_limit=8192,
              supported_generation_methods=['generateContent', 'countTokens'],
              temperature=1.0,
              top_p=0.95,
              top_k=64)
        Model(name='models/gemini-1.5-pro',
              base_model_id='',
              version='001',
              display_name='Gemini 1.5 Pro',
              description='Mid-size multimodal model that supports up to 1 million tokens',
              input_token_limit=1048576,
              output_token_limit=8192,
              supported_generation_methods=['generateContent', 'countTokens'],
              temperature=1.0,
              top_p=0.95,
              top_k=64)
        Model(name='models/gemini-1.5-pro-001',
              base_model_id='',
              version='001',
              display_name='Gemini 1.5 Pro 001',
              description='Mid-size multimodal model that supports up to 1 million tokens',
              input_token_limit=1048576,
              output_token_limit=8192,
              supported_generation_methods=['generateContent', 'countTokens'],
              temperature=1.0,
              top_p=0.95,
              top_k=64)
        Model(name='models/gemini-1.5-pro-latest',
              base_model_id='',
              version='001',
              display_name='Gemini 1.5 Pro Latest',
              description='Mid-size multimodal model that supports up to 1 million tokens',
              input_token_limit=1048576,
              output_token_limit=8192,
              supported_generation_methods=['generateContent', 'countTokens'],
              temperature=1.0,
              top_p=0.95,
              top_k=64)
        Model(name='models/gemini-pro',
              base_model_id='',
              version='001',
              display_name='Gemini 1.0 Pro',
              description='The best model for scaling across a wide range of tasks',
              input_token_limit=30720,
              output_token_limit=2048,
              supported_generation_methods=['generateContent', 'countTokens'],
            temperature=0.9,
            top_p=1.0,
            top_k=None)
      Model(name='models/gemini-pro-vision',
            base_model_id='',
            version='001',
            display_name='Gemini 1.0 Pro Vision',
            description='The best image understanding model to handle a broad range of applications',
            input_token_limit=12288,
            output_token_limit=4096,
            supported_generation_methods=['generateContent', 'countTokens'],
            temperature=0.4,
            top_p=1.0,
            top_k=32)
      Model(name='models/embedding-001',
            base_model_id='',
            version='001',
            display_name='Embedding 001',
            description='Obtain a distributed representation of a text.',
            input_token_limit=2048,
            output_token_limit=1,
            supported_generation_methods=['embedContent'],
            temperature=None,
            top_p=None,
            top_k=None)
      Model(name='models/text-embedding-004',
            base_model_id='',
            version='004',
            display_name='Text Embedding 004',
            description='Obtain a distributed representation of a text.',
            input_token_limit=2048,
            output_token_limit=1,
            supported_generation_methods=['embedContent'],
            temperature=None,
            top_p=None,
            top_k=None)
      Model(name='models/aqa',
            base_model_id='',
            version='001',
            display_name='Model that performs Attributed Question Answering.',
            description=('Model trained to return answers to questions that are grounded in provided '
                         'sources, along with estimating answerable probability.'),
            input_token_limit=7168,
            output_token_limit=1024,
            supported_generation_methods=['generateAnswer'],
            temperature=0.2,
            top_p=1.0,
            top_k=40)
```

```
models = [
    m for m in palm.list_models() if "generateText" in m.supported_generation_methods
]

for m in models:
  print(f"Model Name: {m.name}")
```

```
Model Name: models/text-bison-001
```

```
model = models[0].name
print(model)
```

```
models/text-bison-001
```

Input Prompt

```
prompt = """

Provide a summary of this paragraph by including all the necessary information.
Text: "Johannes Gutenberg (1398 – 1468) was a German goldsmith and publisher who introduced printing to Europe. Hi
Gutenberg many contributions to printing are: the invention of a process for mass-producing movable type, the use
In Renaissance Europe, the arrival of mechanical movable type printing introduced the era of mass communication wh

Summary:"The German Johannes Gutenberg introduced printing in Europe. His invention had a decisive contribution in
Gutenberg major invention was a practical system permitting the mass production of printed books. The printed book

Text: "The Covid-19 pandemic necessitated a global shift to online learning. While researchers have examined the i

"""
```

Summery

```
completion = palm.generate_text(
    model=model,
    prompt=prompt,
    temperature=0.3,
    # The maximum length of the response
    max_output_tokens=800,
)

print(completion.result)
```

```
Summary: The Covid-19 pandemic led to a shift to online learning. This
```

```
#code
```

```
prompt = """

Could you please help me to write code to generate multiples of a number from a given list.

"""
```

```
completion = palm.generate_text(
    model=model,
    prompt=prompt,
    temperature=0.3,
    # The maximum length of the response
    max_output_tokens=800,
)

completion.result
```

```
'```python\ndef generate_multiples(number, list1):\n  """Generates a
list of multiples of a given number from a given list.\n\n  Args:\n
number: The number to generate multiples of.\n    list1: The list to
generate multiples from.\n\n  Returns:\n    A list of multiples of th
e given number from the given list.\n  """\n\n  multiples = []\n  for
```

```
print(completion.result)
```

~~~
```python
def generate_multiples(number, list1):
  """Generates a list of multiples of a given number from a given list.
~~~

```
        Args:
          number: The number to generate multiples of.
          list1: The list to generate multiples from.

        Returns:
          A list of multiples of the given number from the given list.
        """

        multiples = []
        for item in list1:
          multiples.append(number * item)
        return multiples


      print(generate_multiples(3, [1, 2, 3, 4, 5]))
      # [3, 6, 9, 12, 15]
      ```
```

```
prompt = """

I have three visions for India. In 3000 years of our history, people from all over the world have come and invaded
That is why my first vision is that of FREEDOM. I believe that India got its first vision of this in 1857, when we
My second vision for India's DEVELOPMENT, For fifty years we have been A developing nation. It is time we see ours
I have a THIRD vision. India must stand up to the world. Because I believe that, unless India stands up to the wor
I see four milestones in my career:
"""
```
```