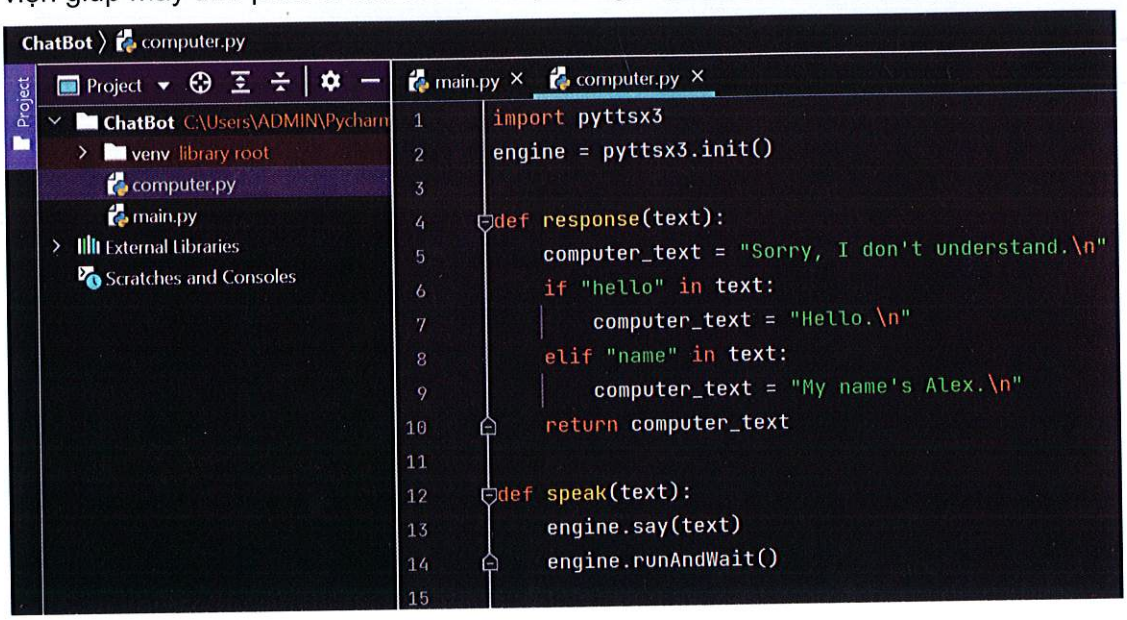
BÀI 4 TÁCH CHƯƠNG TRÌNH THÀNH CÁC MODULE VÀ CHỨC NĂNG PHÁT ÂM THANH ĐỌC LỜI PHẢN HỒI

Trong bài học này, chúng ta sẽ tự mình tách chương trình thành các phần khác nhau và tạo thành thư viện riêng của mình. Ngoài ra, các bạn sẽ được tìm hiểu thư viện giúp máy tính phát ra âm thanh là nội dung một câu văn bất kỳ.

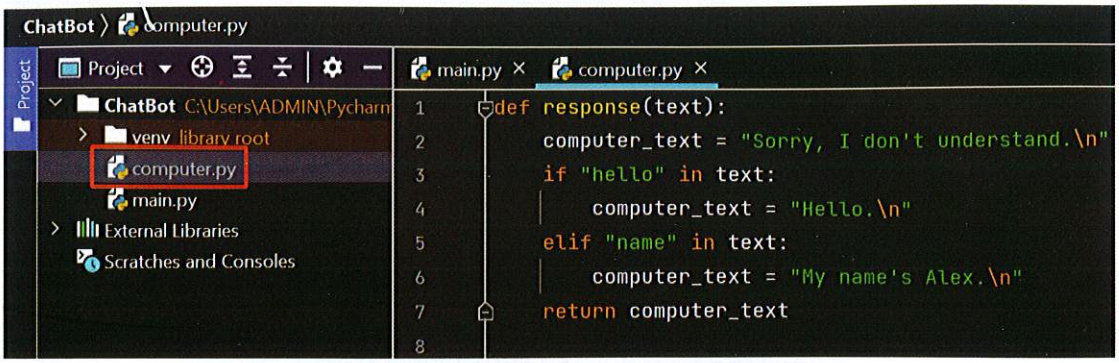


Kết quả của bài học

4.1. Tách chương trình thành các module

Việc chia nhỏ chương trình thành các module sẽ giúp chương trình dễ quản lý và sửa lỗi và phát triển hơn. Trong dự án này, chúng ta sẽ tách việc phản hồi của máy tính thành một thư viện riêng để với các hàm xử lý có đầu vào là văn bản từ người dùng, đầu ra là phải hồi của máy tính, từ đó có thể sử dụng thư viện này trong bất kỳ dự án nào khác.

Đầu tiên, các bạn sẽ tạo một tệp tin python mới bên trong thư mục dự án (ví dụ: chương trình chúng ta đang làm việc trên tập tin **main.py** và tập tin tạo mới là **computer.py**), sau đó lập trình hàm thực hiện việc phản hồi của máy tính (ví dụ: **response()**) có vai trò giống với hàm **computer_response()** đã được lập trình ở bài học trước, nhưng chỉ trả về nội dung phản hồi mà không thực hiện việc chèn vào ô hội thoại.




```
# chương trình trên file computer.py
def response(text):
    computer_text = "Sorry, I don't understand.\n"
    if "hello" in text:
        computer_text = "Hello.\n"
    elif "name" in text:
        computer_text = "My name's Alex.\n"
    return computer_text
```

Trong hàm **computer_response()** của chương trình chính ở tập tin **main.py**, chúng ta sẽ xóa các lệnh kiểm tra và thiết lập lời phản hồi của máy tính, thay vào đó chúng ta sẽ gọi đến hàm **response()** trong tập tin **computer.py** như mô tả trên dòng 11. Tệp tin **computer.py** đóng vai trò là thư viện, chúng ta sẽ import tệp tin này và sử dụng như cách chúng ta import các thư viện khác như mô tả trên dòng 2.

```
# chương trình trên file main.py
import tkinter as tk
import computer

wd = tk.Tk()
wd.title("ChatBot")

def send_text():
    ...
    computer_response(user_text)

def computer_response(text):
    computer_text = computer.response(text)
    text_area.insert(tk.END, 'Computer: ' + computer_text)
```

Các bạn chạy thử sẽ thấy chương trình hoạt động bình thường. Như vậy, chúng ta đã tạo được một thư viện **computer** cho riêng mình, bước đầu hỗ trợ chức năng đưa ra phản hồi của máy tính khi nhận được một chuỗi ký tự đầu vào qua hàm **response()**.

4.2. Lập trình phát âm thanh đọc lời phản hồi sử dụng thư viện pyttsx3

Trong trường hợp chúng ta gặp khó khăn trong việc sử dụng dịch vụ Cloud Text-to-Speech của Google, các bạn có thể cài đặt thư viện hỗ trợ chức năng chuyển đổi văn bản thành giọng nói **pyttsx3.** Sau khi đã thực hành theo phần này, các bạn không cần thực hành theo phần 4.3 sau đây nữa, tuy nhiên các bạn vẫn có thể đọc để tham khảo.

Các bạn cài đặt thư viện **pyttsx3** bằng lệnh **pip install pyttsx3**, sau đó nhập thư viện (dòng 1) và tạo thêm một hàm thực hiện việc đọc với đầu vào là một nội dung văn bản bất kỳ như mô tả ở dòng 9 (ví dụ: **speak()**).

```
   # chương trình trên file computer.py
1  import pyttsx3

2  def response(text):
...    ...
8    return computer_text

9  def speak(text):
10   pass
```

Để thực hiện việc đọc văn bản, chúng ta khởi tạo một đối tượng (ví dụ **engine**) bằng lệnh **init()** (dòng 2), sau đó lập trình thực hiện công việc phát âm thanh đọc một văn bản bằng hàm **say(<văn bản>)** và **runAndWait()** (dòng 11 và 12). Các bạn lưu ý chúng ta có thể phát ra nhiều câu nói bằng nhiều lệnh **say()** khác nhau, nhưng chỉ cần một lệnh **runAndWait()** cuối cùng.

```
   # chương trình trên file computer.py
1  import pyttsx3
2  engine = pyttsx3.init()

3  def response(text):
...    ...
9    return computer_text

10 def speak(text):
11   engine.say(text)
12   engine.runAndWait()
```

Trên chương trình chính, chúng ta thêm lệnh **wd.update()** và gọi hàm **speak()** trong thư viện **computer** để thực hiện việc phát âm thanh đọc sau khi cập nhật nội dung vào ô hội thoại.

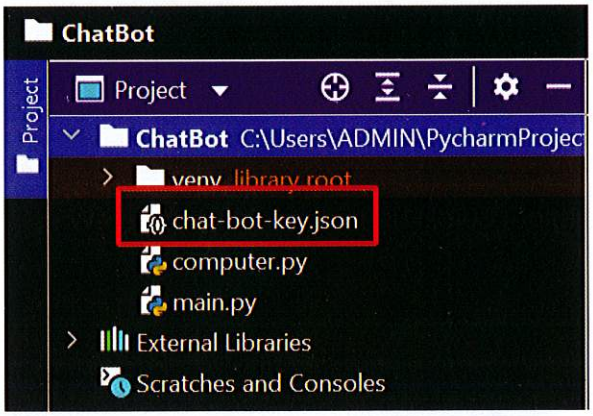
```
    # chương trình trên file main.py
10  def computer_response(text):
11      computer_text = computer.response(text)
12      text_area.insert(tk.END, 'Computer: ' + computer_text)
13      wd.update()
14      computer.speak(computer_text)
```

Như vậy, mỗi khi máy tính phản hồi sẽ kèm theo âm thanh đọc nội dung, các bạn lưu ý khi phát âm thanh thì cửa sổ chương trình sẽ bị đóng băng (giống như bị "đơ", "lag"). Để khắc phục điều này, các bạn có thể tự tìm hiểu thêm về phân luồng (threading) và sử dụng các thư viện hỗ trợ phân luồng như thư viện **threading**.

4.3. Lập trình phát âm thanh đọc lời phản hồi sử dụng dịch vụ Cloud Text-to-Speech của Google

Các bạn lưu ý khi thực hành theo phần này thì bắt đầu sau khi hoàn thành phần 4.1 và bỏ qua phần 4.2. Chúng ta sẽ bổ sung chức năng cho thư viện **computer** sao cho máy tính sẽ phát ra âm thanh là lời đọc câu phản hồi theo cách sử dụng dịch vụ Cloud Text-to-Speech của Google và sử dụng thư viện **google.cloud**.

Đầu tiên, các bạn di chuyển tệp tin khóa riêng tư đã tải về từ Bài 1 vào cùng thư mục với tệp tin **main.py** và đổi tên sao cho ngắn gọn hơn, ví dụ **chat-bot-key.json**.



Tiếp theo, các bạn cài đặt thư viện của Google bằng lệnh `pip install --upgrade google-cloud-texttospeech`, sau đó thêm thư viện để sử dụng (dòng 1) như mô tả ở hình dưới đây. Chúng ta cũng tạo thêm một hàm thực hiện việc đọc với đầu vào là một nội dung văn bản bất kỳ (ví dụ: **speak()**).

```
  # chương trình trên file computer.py
1   from google.cloud import texttospeech

2   def response(text):
...     ...
8     return computer_text
```

```
9  def speak(text):
10   pass
```

Tiếp theo, chúng ta cần thiết lập cho hệ thống biến môi trường bằng cách thêm thư viện os (dòng 2) và thiết lập giá trị là tệp tin chứa khóa riêng tư **chat-bot-key.json** (dòng 3).

```
   # chương trình trên file computer.py
1  from google.cloud import texttospeech
2  import os
3  os.environ['GOOGLE_APPLICATION_CREDENTIALS'] =
                                          'chat-bot-key.json'
4  def response(text):
...    ...
```

Các bạn tiếp tục lập trình thiết lập các cài đặt như chọn ngôn ngữ (trong ví dụ này là tiếng Anh), chọn giọng nói, chọn cách mã hóa âm thanh... như trong các dòng 4, 5, 6 như hình dưới đây. Vì lệnh dài nên ta có thể tách ra thành nhiều dòng.

```
   # chương trình trên file computer.py
1  from google.cloud import texttospeech
2  import os
3  os.environ['GOOGLE_APPLICATION_CREDENTIALS'] =
                                          'chat-bot-key.json'
4  client = texttospeech.TextToSpeechClient()
5  voice = texttospeech.VoiceSelectionParams(
           language_code="en-US",
           ssml_gender=texttospeech.SsmlVoiceGender.NEUTRAL
   )
6  audio_config = texttospeech.AudioConfig(
     audio_encoding=texttospeech.AudioEncoding.MULAW
   )

7  def response(text):
...    ...
```

Thư viện của Google sẽ xuất ra một tệp tin âm thanh trên cùng thư mục với tệp tin main.py là đầu ra của việc chuyển đổi văn bản thành giọng nói. Các bạn thực hiện việc mở tệp tin như hình dưới đây, nếu tệp tin chưa tồn tại sẽ tạo một tệp tin mới, kiến thức về làm việc với tệp tin sẽ được giới thiệu và giải thích trong bài học sau.

```
7  def speak(text):
8    synthesis_input = texttospeech.SynthesisInput(text=text)
9    response = client.synthesize_speech(
                        input=synthesis_input, voice=voice,
                        audio_config=audio_config )
```

```
10      output_file = open("output.wav", "wb")
11      output_file.write(response.audio_content)
```

Để phát âm thanh từ tệp tin **output.wav** đã được tạo ra từ bước trên, ta cần sử dụng hàm **PlaySound()** trong thư viện mới **winsound**. Các bạn bổ sung thư viện (dòng 3) và lệnh **try – finally** như mô tả ở hình dưới đây, ý nghĩa và cách sử dụng của lệnh này sẽ được giải thích trong bài học sau.

```
1   from google.cloud import texttospeech
2   import os
3   import winsound
...   ...

8   def speak(text):
9     synthesis_input = texttospeech.SynthesisInput(text=text)
10    response = client.synthesize_speech(
                          input=synthesis_input, voice=voice,
                          audio_config=audio_config )
11   try:
12     output_file = open("output.wav", "wb")
13     output_file.write(response.audio_content)
14   finally:
15         winsound.PlaySound('output.wav',
                                        winsound.SND_FILENAME)
```

Trên chương trình chính, chúng ta gọi hàm **speak()** trong thư viện **computer** để thực hiện việc phát âm thanh đọc sau khi cập nhật nội dung vào ô hội thoại.

```
    # chương trình trên file main.py
10  def computer_response(text):
11    computer_text = computer.response(text)
12    text_area.insert(tk.END, 'Computer: ' + computer_text)
13    computer.speak(computer_text)
```

Trên chương trình chính, chúng ta thêm lệnh **wd.update()** và gọi hàm **speak()** trong thư viện **computer** để thực hiện việc phát âm thanh đọc sau khi cập nhật nội dung vào ô hội thoại.

```
    # chương trình trên file main.py
10  def computer_response(text):
11    computer_text = computer.response(text)
12    text_area.insert(tk.END, 'Computer: ' + computer_text)
13    wd.update()
14    computer.speak(computer_text)
```

Như vậy, mỗi khi máy tính phản hồi sẽ kèm theo âm thanh đọc nội dung, các bạn lưu ý khi phát âm thanh thì cửa sổ chương trình sẽ bị đóng băng (giống như bị "đơ", "lag"). Để khắc phục điều này, các bạn có thể tự tìm hiểu thêm về phân luồng (threading) và sử dụng các thư viện hỗ trợ phân luồng như thư viện **threading**.

Đến đây chúng ta đã hoàn thành lập trình phát âm thanh đọc lời phản hồi sử dụng dịch vụ Cloud Text-to-Speech của Google, các bạn có thể thử lập trình theo cả 2 cách trong phần 4.2 và 4.3, sau đó tự mình rút ra nhận xét về sự khác biệt trong kết quả.

Tóm tắt lý thuyết và bài tập thực hành

Trong bài học này, chúng ta đã tìm hiểu về cách tạo một thư viện riêng và lập trình chức năng phát âm thanh phản hồi thông qua thư viện **pyttsx3** thông qua các hàm **say()** và **runAndWait()**, ngoài ra các bạn cũng đã được tìm hiểu cách sử dụng dịch vụ Cloud Text-to-Speech của Google.

Bài tập 1. Hãy thử bỏ lệnh **wd.update()** ở dòng 13 và chạy thử, quan sát kết quả hoạt động và giải thích vai trò của lệnh này.

Bài tập 2. Hãy lập trình để khi bắt đầu chương trình, máy tính sẽ đưa ra lời chào trước, ví dụ "Nice to meet you again".