Python JARVIS Text to Speech

Contents

Time required: 60 minutes

This inspiration for this series of tutorials is from:

https://www.freecodecamp.org/news/python-project-how-to-build-your-own-jarvis-using-python/

The JARVIS Project and The Road to OOP

This is a fun tutorial that is a first step to building your own JARVIS/FRIDAY (Iron Man) personal assistant.

This project demonstrates how to use Agile development to create a program. We will have 3 iterations after Pseudocode.

- Pseudocode/TODO

- Top to bottom script

- OOP

When developing a program, start small and simple. Get that to work, add more to it. That is the pattern we will follow with this tutorial.

We want to be able to take any Python script that we find and transform it into an OOP program. This makes it much easier to move into a GUI or Web Based application. OOP makes the core code more modular and portable. The only change we make is the interface.

Requirements Update

Requirements for tutorials and assignments are mean to be minimum guidelines. As long as you meet the requirements, you are free to explore, expand and modify your programs. That is where a large part of higher-level learning takes place. You are beyond the need for step-by-step directions for everything.

Comment your code so I or anything else can understand and carry on with your program.

Take the concepts you have learned; start creating your own unique, creative versions of the assignments.

pyttsx3

Our first step to building a DIY Iron Man suit is to get our computer to speak to us.

pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline. An application invokes the pyttsx3.init() factory function to get a reference to a pyttsx3. Engine instance. it is a very easy to use tool which converts the entered text into speech.

The pyttsx3 module supports two voices in Windows. One is female and the second is male which is provided by "sapi5" for windows.

There are a lot of possibilities for creative programs using this library.

Pyttxs3 supports three TTS (Text to Speech) engines:

- sapi5 – SAPI5 on Windows

- nsss – NSSpeechSynthesizer on Mac OS X

- espeak – eSpeak on every other platform

Tutorial 1: Text to Speech CLI

1. Install the **pyttsx** module: **pip install pyttsx3**

2. Create this short test script to test the new library with minimal functionality and hard coded values.

```python
"""
    Name: text_to_speech_cli_1.py
    Author:
    Created:
    Purpose: Render text into speech
    This library has many modules with which you can try
    changing the voice, volume, and speed rate of the audio.
    https://pypi.org/project/pyttsx3/
    https://pyttsx3.readthedocs.io/en/latest/
"""
# Linux:   pip3 install pyttsx3
# Windows: pip install pyttsx3
import pyttsx3

# init function creates an engine
# instance/object for speech synthesis
engine = pyttsx3.init()

# Pass text to engine.say method
engine.say("Hello, how may I help you?")

# run and wait method processes the voice
engine.runAndWait()
```

Example run:

In a male voice, your computer should say: "**Hello, how may I help you?**"

Tutorial 2: Text to Speech

Let's add some options and get input from the user. It is a good idea not to make too many changes at once.

When you are developing a new program, it is a good idea to keep a copy of each iteration. If something goes wrong, you can go back to a known working version to see where you went wrong.

```python
"""
    Name: text_to_speech_cli_2.py
    Author:
    Created:
    Purpose: Render text into speech
    This library has many modules with which you can try
    changing the voice, volume, and speed rate of the audio.
    https://pypi.org/project/pyttsx3/
    https://pyttsx3.readthedocs.io/en/latest/
"""
# pip install pyttsx3
import pyttsx3

# ---------------- INITIALIZE TEXT TO SPEECH ENGINE -------------------- #
# init function creates an engine
# instance/object for speech synthesis
engine = pyttsx3.init()

# ----------------- VOICE PROPERTIES CONSTANTS ------------------------- #
RATE = 175      # integer default: 200 words per minute
VOLUME = .7  # float default: 1.0, range: 0.0-1.0 inclusive
VOICE = 1     # integer default: 0 for David (male), set 1 for Zira (female)

# ------------------- SET VOICE PROPERTIES --------------------------- #
# Set properties before you add items to say
engine.setProperty("rate", RATE)        # Speed for words per minute
engine.setProperty("volume", VOLUME)  # Volume 0.0-1.0

# Retrieves all available voices from your system into a list
voices = engine.getProperty("voices")

# You can print the voices list to see the voices on your system
for voice in voices:
    print(voice)

# Set voice property
engine.setProperty("voice", voices[VOICE].id)

# Get input from user
speak = input("What would you like me to say? ")

# Queue up input from user
engine.say(speak)

# Flush the say() queue and play the audio
engine.runAndWait()
```

Assignment 1: Text to Speech Interactive OOP

Object Oriented Programming allows us to easily expand our program. This is the first program in a series of programs to create a JARVIS personal assistant of your own.

In previous version, we used hard coded literal strings. In this version we are using string variables to pass what we want to say to the speak method.

Here is a program scaffold to get you started on your way to being Iron Man!

Copy and paste this starter code to get started.

```python
"""
    Name: text_to_speech_cli_3.py
    Author:
    Created:
    Purpose: Render text into speech
    This library has many modules with which you can try
    changing the voice, volume, and speed rate of the audio.
    https://pypi.org/project/pyttsx3/
    https://pyttsx3.readthedocs.io/en/latest/
"""
# pip install pyttsx3
import pyttsx3


class TextToSpeech():
    def __init__(self):
        # -------------- INITIALIZE TEXT TO SPEECH ENGINE ----------- #
        # init function creates an engine
        # instance/object for speech synthesis
        # initialize self.engine

        # Your code here


        # --------------- VOICE PROPERTIES CONSTANTS ---------------- #
        # The constants stay the same

        # Your code here


        # ------------------ SET VOICE PROPERTIES ------------------- #
        # Set properties before you add items to say
        # Use self.engine to set voice properties
```

```python
        # Your code here

    def speak(self, speak):
        # self.engine say

        # Your code here

        # Program will not continue execution until
        # all speech conversion has completed

        # Your code here

def main():
    # Create text to speech object
    text_to_speech = TextToSpeech()

    while True:
        # Get input from user
        speak = input("What would you like me to say? ")

        # Queue up things to say
        text_to_speech.speak(speak)

main()
```

GitHub Presence

One of our unofficial projects this semester is to work on building everyone's personal GitHub repository.

GitHub is a presence and resume for programmers and their skills. Everything in my GitHub there is either my creation or a fork of something from GitHub.

www.github.com/itinstructor

Give credit where credit is due. Look at the copyright. Don't publish someone's else's work as your own.

I use Creative Commons copyright for any projects I publish on GitHub.

https://creativecommons.org/licenses/by/3.0/us/

Assignment Submission

1. Attach the pseudocode.

2. Attach the program files.

3. Submit in Blackboard.