Grow with Google

Technical Writing Assessment

by Brian Hogan, bphogan@syr.edu, 3/4/22

Part I

Congratulate yourself on making the decision to explore programming. The world desperately needs programmers with your skillsets. Programming is about realizing the art of the possible through voice, image, sounds, colors, mathematics, and science. Python is software language computers use to "do things" such as make video monster battle games or take videos of Mars on a robotic helicopter. Computer hardware provides a powerful means to translate the world around us and explore the *art of the possible*.

This exercise will translate something said into text and audio. Start by **import**ing software libraries with powerful code containers, called ***objects***, that perform ***code instructions*** that tell a computer's microprocessor what, when, and how to do something. An object is often based in hundreds of lines of code that form a ***procedure***, or method, to accomplish tasks. Speaking into the computer microphone ***hardware*** is possible because the **import pyaudio** library has a listening ***object*** with ***procedures*** that generate voice ***data***. The data is used by a *Google speech recognition library* with a ***module*** to translate voice data into printed text. It has a different ***module*** to create new data, i.e. an audio file. Let us get started and code in Python.

words: 199

```python
import os
os.chdir('C:\\Users\\17574\\Desktop')
import speech_recognition as sr
import pyaudio
with sr.Microphone() as source:
    print("Ready? Say something quick")
    myWords = sr.Recognizer().listen(source)
    print("You Said...: "+ sr.Recognizer().recognize_google(myWords))
with open("myAudio.wav", "wb") as file_:
    file_.write(myWords.get_wav_data())
from playsound import playsound
playsound('myAudio.wav')
```

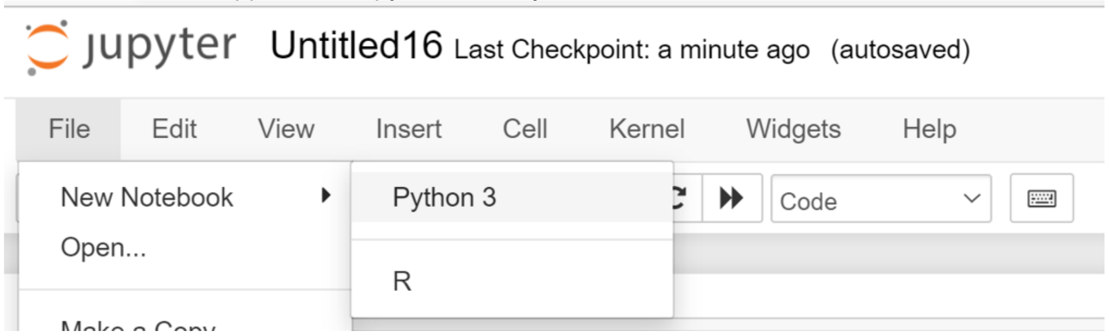
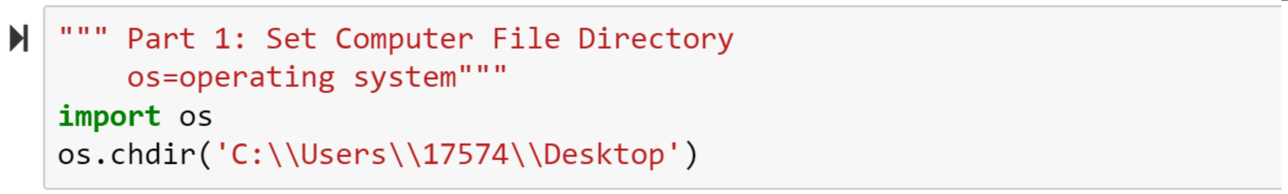
Part 2:

Objectives:

1. Demonstrate how to create a small Python program, called a ***script***, and generate speech to text and text to audio results.
2. Challenge a user to replicate proper syntax, indenting, and other ***coding idioms*** to ensure programs run as intended.
3. Educate on basic data encoding where ***binary*** (1 or 0) is used for pictures/voice and ***nonbinary*** (byte/collations) is for text.
4. Educate on how libraries simplify program feature engineering making the art of the possible a far less daunting task.

ID	Purpose	Code
1	(Library) computer features	`import os`
2	• **instruction where to store a file**	`os.chdir('C:\\Users\\17574\\Desktop')`
3	(Library) speech generation	`import speech_recognition as sr`
4	(Library) audio generation	`import pyaudio`
5	**>method to activate microphone**	`with sr.Microphone() as source:`
6	**>ask a user to speak**	`    print("Ready? Say something quick")`
7	**>translate what user speaks**	`    myWords = sr.Recognizer().listen(source)`
8	**>display what user speaks**	`    print("You Said...: "+ sr.Recognizer().recognize_google(myWords))`
9		`with open("myAudio.wav", "wb") as file_:`
10	**>>create an audio file**	`    file_.write(myWords.get_wav_data())`
11	(Library) audio generation	`from playsound import playsound`
12	• **play audio file back to user**	`playsound('myAudio.wav')`

Scenario 1: Generate a working program in a Python integrated development environment (IDE) such as Anaconda. The following example uses the Jupyter notebook program as part of the Anaconda Install.

Step	Task
1.	• Open Jupyter notebook and select File\New Notebook \ Python 3 • Python 3 is needed for support of the pyaudio library  jupyter Untitled16 Last Checkpoint: a minute ago (autosaved) File Edit View Insert Cell Kernel Widgets Help New Notebook Open... Python 3 R Code
	• Use the "+" symbol to add separate instruction statements • Use this to make sure each code section runs • Once completed will run like a pro in the last section for a seamless interaction • Desired outcome: Speak vocally at computer => see text on screen => hear audio from machine
2.	

```
""" Part 1: Set Computer File Directory
    os=operating system"""
import os
os.chdir('C:\\Users\\17574\\Desktop')
```

3.

```python
""" Part 2: Set Google Speech Recognition
            and Microphone Library Functions """
import speech_recognition as sr
import pyaudio
```

4.

Use of “with <> as <>:

- With specifies a module feature and what action is taken upon on.
- Microphone is a source input
- Variable mywords is assigned what words spoken with,
    - The outcome of the the google speech recognizer feature .listen

```python
""" Part 3: Ask user to same something
    use Google speech to parse words"""
with sr.Microphone() as source:
    print("Ready? Say something quick")
    myWords = sr.Recognizer().listen(source)
    print("You Said...: "+ sr.Recognizer().recognize_google(myWords))
```

5.

```python
"""Part 4: Encode words into audio file
    audio data is binary so add 'wb'
    for 'write binary data (1 or 0)"""
with open("myAudio.wav", "wb") as file_:
    file_.write(myWords.get_wav_data())
```

6.

```python
"""Part 5: Import a generic microphone module """
from playsound import playsound
playsound('myAudio.wav')
```

Run like a Pro

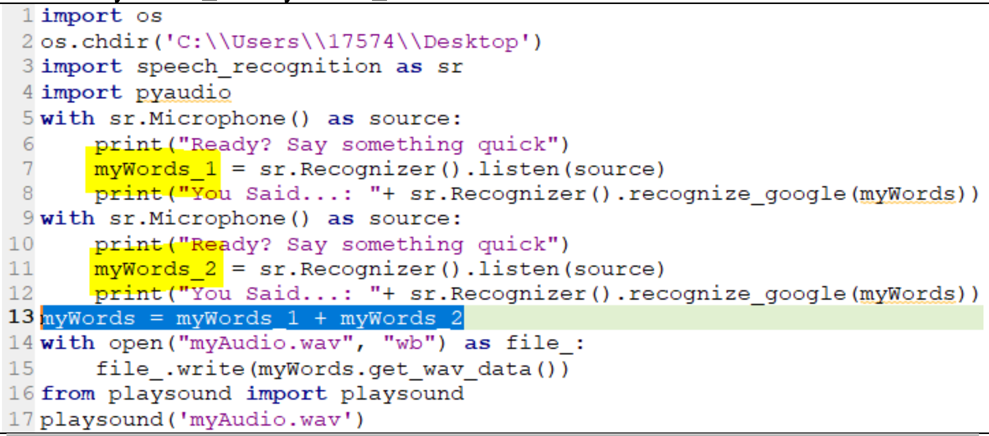
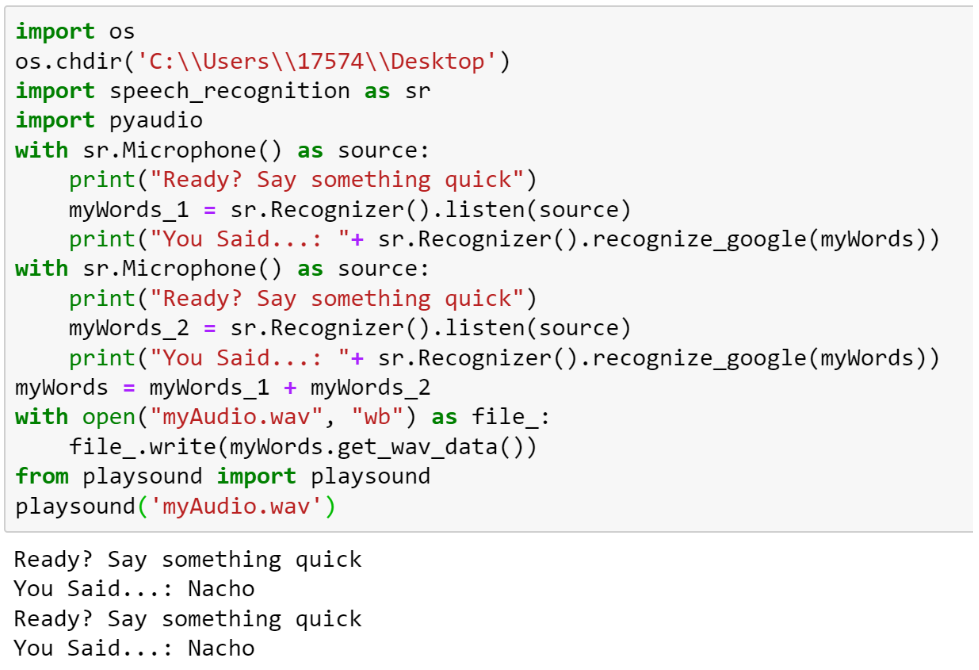
```python
""" Run like a Pro """
import os
os.chdir('C:\\Users\\17574\\Desktop')
import speech_recognition as sr
import pyaudio
with sr.Microphone() as source:
    print("Ready? Say something quick")
    myWords = sr.Recognizer().listen(source)
    print("You Said...: "+ sr.Recognizer().recognize_google(myWords))
with open("myAudio.wav", "wb") as file_:
    file_.write(myWords.get_wav_data())
from playsound import playsound
playsound('myAudio.wav')
```

```
Ready? Say something quick
You Said...: I like cake
```

Scenario 2: Expand code requiring 2 audio requests but deliver a single audio outcome file

Hint: The trick of this scenario is to create 2 separate myWords variables.

- In Python variables are either implicitly or explicitly declared.
- Code line 7 “my Words” is an implicit declaration as its type is not declared, such a character (char) or number
- Add a “_1” to the variable and then duplicate code lines 5-8 with a second variable myWords_2
- Finally, combine the myWords_1 with myWords_2 into myWords to deliver the audio output

Step	Task
1.	Duplicate code rows 5 to 8
2.	Paste after row 9 so new line is one row 9
3.	Change names of myWords to: myWords_1 and myWords_2
4.	Add a new line of code on line 13 • myWords = myWords_1 + myWords_2
Result	
Jupyter Notebook	

```
1 import os
2 os.chdir('C:\\Users\\17574\\Desktop')
3 import speech_recognition as sr
4 import pyaudio
5 with sr.Microphone() as source:
6     print("Ready? Say something quick")
7     myWords_1 = sr.Recognizer().listen(source)
8     print("You Said...: "+ sr.Recognizer().recognize_google(myWords))
9 with sr.Microphone() as source:
10    print("Ready? Say something quick")
11    myWords_2 = sr.Recognizer().listen(source)
12    print("You Said...: "+ sr.Recognizer().recognize_google(myWords))
13 myWords = myWords_1 + myWords_2
14 with open("myAudio.wav", "wb") as file_:
15     file_.write(myWords.get_wav_data())
16 from playsound import playsound
17 playsound('myAudio.wav')
```

```
import os
os.chdir('C:\\Users\\17574\\Desktop')
import speech_recognition as sr
import pyaudio
with sr.Microphone() as source:
    print("Ready? Say something quick")
    myWords_1 = sr.Recognizer().listen(source)
    print("You Said...: "+ sr.Recognizer().recognize_google(myWords))
with sr.Microphone() as source:
    print("Ready? Say something quick")
    myWords_2 = sr.Recognizer().listen(source)
    print("You Said...: "+ sr.Recognizer().recognize_google(myWords))
myWords = myWords_1 + myWords_2
with open("myAudio.wav", "wb") as file_:
    file_.write(myWords.get_wav_data())
from playsound import playsound
playsound('myAudio.wav')
```

```
Ready? Say something quick
You Said...: Nacho
Ready? Say something quick
You Said...: Nacho
```

Quiz:

Q1) What does “wb” stand for and why was it used in creation of the audio file?

> Answer: wb = write binary. While text data is coded in nonbinary constructs of collations or characters, audio and image data are encoded in binary format (1 or 0). (29 words)

Q2) What is the main difference between the use of double quotations (“ “) and single quotations (‘ ‘)?

> Answer: in python double quotations denotes text strings and code methods or instructions require string values. Single quotes is reserved for inserting paths or file names into instruction command consistent with underlying operating system language. (35 words)

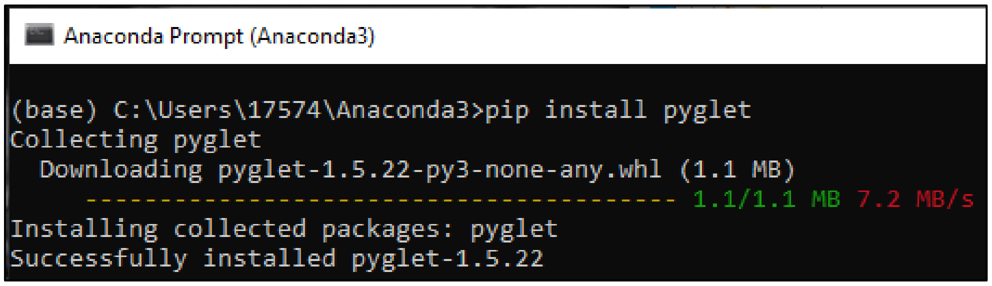
Q3) [extra points] what is meant by phrase ‘coding idioms’ in the 2nd objective statement at start of Part 2?

> Answer: coding or programming idioms is a semantic structure of a programming language. A semantic structure represents a combination of indentation, parenthesis, brackets, single/double quotes, and anaphora combinations. (28 words)

# Words: 590 – (answer words)(29+35+28)= 498

Appendix – Script for running with a Command Prompt

Use the embedded file to run on a terminal or command prompt with Python 3 installed. Ensure to install Python libraries using pip automation on a command prompt. For example:



```
Anaconda Prompt (Anaconda3)

(base) C:\Users\17574\Anaconda3>pip install pyglet
Collecting pyglet
  Downloading pyglet-1.5.22-py3-none-any.whl (1.1 MB)
     ---------------------------------------- 1.1/1.1 MB 7.2 MB/s
Installing collected packages: pyglet
Successfully installed pyglet-1.5.22
```

For more information see, https://realpython.com/what-is-pip/ .

Command Instruction	File	Pip Install Files & Sequence
py C:\Users\17574\Desktop\my_program.py	my_program.py	pip install SpeechRecognition pip install playsound pip install pipwin pipwin install pyaudio

```
# Command Prompt instructions
# py C:\Users\17574\Desktop\my_program.py

""" Part 1: Import Library Facilitators"""
import os
import pyaudio
import speech_recognition
from playsound import playsound

""" Part 2: Set library features """
import os
os.chdir('C:\\Users\\17574\\Desktop')
import speech_recognition as sr  #google speec recogn.
r = sr.Recognizer() # Initialize speech recognizing

""" Part 3: Talk and Record into Computer Microphone """
#import pyaudio
with sr.Microphone() as source:
    print("Say something quick !")
    audio_text = r.listen(source)
    print("Thank you, just a second")
    try:
        # using google speech recognition
        print("Text: "+r.recognize_google(audio_text))
    except:
         print("Sorry, I did not get that")
```

```
""" Part 4: Have Computer Play Back Message"""
with open("mySound.wav", "wb") as f_ile:
    f_ile.write(audio_text.get_wav_data())
#from playsound import playsound
playsound('C:\\Users\\17574\\Desktop\\mySound.wav')

#---------------Additional Information----------------------#
"""
Google search => what does 'wb' mean in python?
while binary mode must be used when writing non-text files like images.
The wb indicates that the file is opened for writing in _
binary mode.
When writing in binary mode, Python makes no _
changes to data as it is written to the file. In text mode _
(when the b is excluded as in just w or when you specify text_
 mode with wt), however, Python will encode the text based _
on the default text encoding.
Additionally, Python will _
convert line endings (\n) to whatever the platform-specific_
line ending is, which would corrupt a binary file like an _
exe or png file. Text mode should therefore be used when _
writing text files (whether using plain text or a text-based_
format like CSV),while binary mode must be used when writing _
non-text files like images.
References:
https://stackoverflow.com/questions/2665866/what-does-wb-mean-in-this-code-using-python
https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files
https://docs.python.org/3/library/functions.html#open
edited Oct 1, 2019
"""
```