# Open Window Spanish Mad Libs Coding Project

Max Carey, Open Window Spanish Department 3/6/2017

Welcome to the coding section of the Spanish Mad Libs project. This tutorial is for those of you who want to work independently and/or cannot make the Lab Sessions. First, you will build the following sample Mad Lib:

Esto **[INSERT_VERB_SER]** mi primer Mad Lib. **[INSERT_POSSESSIVE_ADJECTIVE]** nombre es Max y soy muy **[INSERT_MSN]**. Me gustan las **[INSERT_FPN]**.

So when someone plays the Mad Lib, they could get the following:

Esto **es** mi primer Mad Lib. **Mi** nombre es Max y soy muy **bajo**. Me gustan las **pizzas**.

Let's walk step-by-step to build this Mad Lib. Then you will build your own Mad Lib.

## 1. Import the madlib.py file

The very first line in our program needs to install some code that has already been provided to us (known as the library code). The library code is on a file called madlib.py that is in our current working directory.

Listing 1: Put this at the top of your program:

```
import madlib
```

## 2. Enter your verb forms as Python Dictionaries

The next thing we need to do is create a Python Dictionary for the verb "ser" (eventually you will do this for each verb in your Mad Lib). A Python Dictionary is a collection of key-value pairs. In the dictionary below, the keys are: "INF", "1PS", "2PS", etc. The values are the corresponding verb forms: ser, soy, eres, etc.

Listing 2: Put this dictionary of "ser" below your import line

```
import madlib

ser = {
    "INF": ["ser"],
    "1PS": ["soy"],
    "2PS": ["eres"],
    "3PS": ["es"],
    "1PP": ["somos"],
    "2PP": ["sois"],
    "3PP": ["son"],
}
```

Starting on line 3, we are creating a Python Dictionary called "ser". Notice that we are only using the present tense and the infinitive forms of the verb "ser": The definition of our keys are represented in the table below:

| Key | Definition |
|---|---|
| INF | infinitive |
| 1PS | 1st person singular |
| 2PS | 2nd person singular |
| 3PS | 3rd person plural |
| 1PP | 1st person plural |
| 2PP | 2nd person plural |
| 3PP | 3nd person plural |

## 3. Create variables for the different words you have chosen (VERBS)

Now, we need to create a variable called "verbOne". A variable is like a box that you put a value into. [1]

Listing 3: Add this to the bottom of your code

```
verbOne = ser["3PS"][0]
```

In the above snippet, we are "pulling" a value from the dictionary that we created in the last section and putting it inside our new variable, verbOne. By entering the key "3PS", we chose "es" to be the value that occupies our variable.

Note: The [0] allows us to get the variable without the quotation marks and brackets.

## 4. Create variables for the different words you have chosen (Possessive Adjectives)

Now we need to get our possessive adjectives into variables. We will do this almost the same way that we got our variable verbOne. However, the possessive adjective dictionary has already been created and is located in our library code. Notice how we can access it with the code below:

[1] In Python, this process is actually known as variable assignment

Listing 4: Add this to the bottom of your code

```
possAdj = madlib.possessiveAdj["1PS_SO"][0]
```

The **madlib.** notation is what allows us to access the library code. If you opened up the madlib.py you would see the following PYthon dictionary:

Listing 5: Dictionary for possessive adjectives (located in library code)

```
1  possessiveAdj = {
2      "1PS_SO": ["Mi"],
3      "1PS_PO": ["Mis"],
4      "2PS_SO": ["Tu"],
5      "2PS_PO": ["Tus"],
6      "3PS_SO": ["Su"],
7      "3PS_PO": ["Sus"],
8      "1PP_MSO": ["Nuestro"],
9      "1PP_MPO": ["Nuestros"],
10     "1PP_FSO": ["Nuestra"],
11     "1PP_FSO": ["Nuestras"],
12     "2PP_MSO": ["Vuestro"],
13     "2PP_MPO": ["Vuestros"],
14     "2PP_FSO": ["Vuestra"],
15     "2PP_FPO": ["Vuestras"],
16     "3PP_SO": ["Vuestras"],
17     "3PP_PO": ["Vuestras"],
18 }
```

One important thing to note is the naming convention for the keys in this dictionary. The three characters to the left indicate person and number of the subject (just like the verbs). The characters to the right of the underscore indicate the number and gender of the object.

For example, on line two, the key "1PS_SO" means 1st person singular, singular object.

## 5. Create variables for the different words you have chosen (Nouns and adjectives)

Let's get our nouns and our adjectives! To do this we will be using a function called getWord() that is inside our library code, so don't forget to use the **madlib.** notation.

Listing 6: Add this to the bottom of your code

```
adjOne = madlib.getWord("msa")
nounOne = madlib.getWord("fpn")
```

What do you think "msa" and "fpn" stand for? We are telling the user that we want a masculine singular adjective and a feminine plural noun!

Try running the program and see what happens!

## 6. Putting it all together: Creating our Mad Lib

It's time to create the Mad Lib now. We need to create a variable called myStory and set it equal to our Mad Lib skeleton. Inside the skeleton, we will put a bunch of these bracket paris: {} that represent the blank line in a traditional, paper Mad Lib. Inside of these brackets, we will place our variables.

Listing 7: Add this to the bottom of your code

```
1 myStory = ('Esto {} mi primer Mad Lib: {} nombre es Max, y soy muy {}. Me gustan las {}.'.format(verbOne,possAdj,adjOne,nounOne))
2
3 print(myStory)
```

Do you see what goes inside the .format()? Our variables! The .format() places the variables inside our skeleton in the order that the {}s appear. Also don't forget to print your Mad Lib (seen on line 3 above)

## 7. Getting Google's Text-To-Speech to Read your Mad Lib

Now for the fun part! We are going to use a library function called getMP3() to get Google's Text To Speech to read our Mad Lib.

Listing 8: Add this to the bottom of your code

```
madlib.getMP3(myStory)
```

We are passing our story into the getMP3() function.

## 8. Making your Mad Lib program a function

The next thing we need to do is wrap our code inside a function. We need to declare our function using the **def** keyword and indent everything that occurs inside our function. Observe in the example below:

Listing 9: Your program should look like this now

```
def myMadLib():
 2      verbOne = ser["3PS"][0]
 3      adjOne = madlib.getWord("msa")
 4      nounOne = madlib.getWord("fpn")
 5      possAdj = madlib.possessiveAdj["1PS_SO"][0]
 6      myStory = ('Esto {} mi primer Mad Lib: {} nombre es Max, ' \
 7                 'y soy muy {}. Me gustan las {}.'.format(verbOne,possAdj,↩
                      adjOne,nounOne))
 8      print(myStory)
 9      madlib.getMP3(myStory)
10
11  myMadLib()
```

Don't forget to call your function. That's what we are doing on line 11.

## 9. Give your data to the getRandomWord() function

When you run your program and play your Mad Lib, enter random when the program prompts you for specific nouns and adjectives. Does it work? No. This is because the program doesn't have any lists of words to draw from. You will have to make use of the lists of words that you investigated in Part 1 and use your knowledge of Spanish gender and number to create sublists. Fortunately, the word lists are located on the library code so we can access them with the **madlib.** notation.

Listing 10: Add this to the top of your code (below the import line)

```
1  fsaDict = madlib.adj_a + madlib.adj_e + madlib.adj_l
```

On line 1, we create a list called fsaDict, "fsa" stands for feminine singular adjective. Therefore, we add the lists adj_a, ad j_e, and adj_l to our new list (fsaDict). Remember: these words end in the suffixes -a (loca), -e (inteligente), and -l (social), respectively. Don't forget that even though you may consider adjectives that end in -l and -e to be gender neutral, they still count as either masculine or feminine because they take the gender of the noun they modify. Technically, adjectives don't don't have grammatical gender.

In total, you will need 8 "sub-dictionaries": fsaDict, fpaDict, msaDict, mpaDict, msnDict, mpnDict, fsnDict, and fpnDict.

After creating your dictionaries you will need to "activate" them. We do this by passing our newly created lists into the setLists() function of the library code. Go ahead and copy the following code to your Mad Lib.

Listing 11: Copy this code(below the creation of your lists)

```
madlib.setLists(fsaDict, fpaDict, msaDict, mpaDict, msnDict, mpnDict, ↩
    fsnDict, fpnDict)
```

It is a good idea to copy and paste this line because you need the arguments to be the same order as they appear in this tutorial.

## 10. Make your own Mad Lib

Follow the same steps above to create and implement your own Mad Lib.

**One thing that is important to remember:** For possessive adjectives and verbs, the user doesn't enter them into the computer. Instead, you (the programmer) enters them in. On the other hand, for nouns and adjectives, the user does enter them. This was done intentionally so that you can complete Part III.