Python Challenge #1

In Python, we can use *lists* to store related items together in a single place. Two examples are:

```
tanzanian_election_years = [1995, 2000, 2005, 2010, 2015]
tanzanian_presidents = ["John Magufuli", "Jakaya Kikwete", "Benjamin Mkapa", \
                        "Ali Hassan Mwinyi", "Julius Nyerere"]
```

1. On a computer or tablet, write a Python program that prints out each year that Tanzania has had a general election. In Python, you can loop over the items of a list like this:

   ```
   for item in list:
       # do something with item
   ```

2. Using your answer for #1, change it slightly so that it *only* prints out the years in which Tanzania has had a general election *after the year 1999.*

3. Sometimes we want to work with two lists at the same time. Say we are given the following list of Tanzanian vice presidents:

   ```
   tanzanian_vice_presidents = ["Samia Suluhu", "Mohamed Gharib Bilal", \
                                "Omar Ali Juma", "Cleopa Msuya", "Aboud Jumbe"]
   ```

   and we want to match each vice president with the president he or she served with. We can use Python's *zip()* function to do this. To illustrate how *zip()* works, type the following into your Python interpreter (note: don't type the ">>>", that is printed by the Python interpreter itself):

   ```
   >>> fruits = ["apple", "grape", "blueberry"]
   >>> colors = ["green", "purple", "blue"]
   >>> zip(fruits, colors)
   [('apple', 'green'), ('grape', purple'), ('blueberry', 'blue')]
   ```

   Notice how Python matched each of the fruits in our first list with its corresponding color in the second list. The *zip()* function itself returns a list, which we can use in our own *for* loops.

   On your computer or tablet, write a Python program that zips together the `tanzanian_presidents` list with the `tanzanian_vice_presidents` list. Then use a *for* loop to print out each pair on a separate line.

4. A *method* is a Python function that is called on a specific value or variable. For instance, the *startswith(str)* function is a method that can be used to see if a Python string starts with another string; try typing the following into your Python interpreter:

   ```
   >>> 'Leo asubuhi'.startswith('Leo')
   True
   >>> 'Leo asubuhi'.startswith('leo')
   False
   >>> str = 'Kesho asubuhi'
   >>> str.startswith('Kesho')
   True
   >>> str.startswith('kesho')
   False
   ```

   Using the `tanzanian_vice_presidents` list, write a program that prints out only those Tanzanian vice presidents whose first name is Samia or Omar.

5. We often want to *sort* a list before working with it. Python makes it very easy to do this. We can use the *sort()* method on any list to sort the list's elements in ascending order. Try this in your Python interpreter:

```
>>> colors = ["green", "purple", "blue"]
>>> colors.sort()
>>> print colors
['blue', 'green', 'purple']
```

Write a Python program that first sorts the list of Tanzanian presidents, then prints out only those presidents whose name starts with the letter "J".

6. On your computer or tablet, write a Python program that defines the following list:

```
people = ["Juma", "Hamisi"]
```

and then add all of the names in the `tanzanian_presidents` and `tanzanian_vice_presidents` lists to the `people` list. Finally, print the `people` list to the screen; it should now contain the two original names, plus those of the five Tanzanian presidents and the five Tanzanian vice presidents. Your program should use a Python method to do this; before you write code for this problem, read about the different methods that you can use on Python lists here:

https://docs.python.org/2/tutorial/datastructures.html#more-on-lists