Let's build a real program! The Chinese Zodiac

1. Plan the Program

For us this means understanding the problem and being very clear what the steps are towards creating a solution. A step is really any part of the program for which we should carry out independent testing to make sure that it works. This will make the construction of our programs *iterative*.

To understand what it is you are being asked to do download this file from the GitHub repository for this class: **http://bit.ly/1OCXLh9**. Put it in a directory you are comfortable with and open it in iPython/Jupyter notebooks. Or you can follow the link and then copy and paste into the notebook.

This file will contain the following instructions as a multi-line comment (note that everything is wrapped inside triple double quotes).

```
"""
While taking a class on the culture of China, you have learned about the Chinese
zodiac in which people fall into 1 of 12 categories, depending on the year of
their birth. The categories, numbered 0 to 11, correspond to the following
animals:

(0) monkey
(1) rooster
(2) dog
(3) pig
(4) rat
(5) ox
(6) tiger
(7) rabbit
(8) dragon
(9) snake
(10) horse
(11) goat

Those who believe in this zodiac think that the year of a person's birth
influences both their personality and fortune in life.

Your task is to build a program that will ask a user for their birth year and
tell them their zodiac sign.  If the user does not enter a number that can be
interpreted as a year then an error message must be shown and the user given
another chance.  If the user types "quit" then the program halts.

For extra credit: save each year that is input to a file and print a chart
showing how many of each type or animal have been returned.

To find your zodiac sign, divide the year of your birth by 12. The remainder
then determines your sign. For example: The remainder when we divide 1985 by 12,
is 5; therefore, a person born in 1985 is an ox according to the Chinese zodiac.
"""
```

So, what we need to do now is add comments that will act as placeholders/sign-posts/instructions as we build the program. So, we add the following to the top of the file:

```
#Here we are planning the components of the program.

#Open the zodiac file

#Load the file

#Ask user for input (year)

#Take year and use as a conditional

#Return horoscope

#Repeat

#Below is a description of the task
```

2. Open the File

We need to get a set of zodiac descriptions. We'll use the file at **http://bit.ly/1KiRoYw** since it is already set up for our purposes. Collect it however you would like and put it in the same directory as the zodiacTool-0.py file that you are working with.

This is how we'll open the file:

```
#Open the zodiac file

zodiacText = open('zodiacDescriptions.txt')
```

If we run this though apparently nothing happens. Here too "silence is golden" and we get what we ask for. What we want to do is print out the contents of the file. Opening the file produces a "pointer" that is stored in a variable called "zodiacText". We can use this pointer and its default return value (lines in the file) to print out the file content.

```
#Open the zodiac file

zodiacText = open('zodiacDescriptions.txt')
for line in zodiacText:
    print(line)
```

Of course, if we open something it is a good idea to explicitly close it and so we do just that at the very end.

```
#Repeat

zodiacText.close()

#Below is a description of the task
```

Here we see our first "method". Methods are functions attached to specific data types that are called with the dot-method format seen here. Methods are often empty functions, hence the empty parentheses, but this is not necessary. You can see what methods are available in some interpreters by typing the variable name, adding a period immediately after, and then pressing TAB.

3. Load File into a List

We don't actually want to print out the content of the zodiac program for anything other than testing so we'll just comment out the loop and print lines.

```
#Open the zodiac file

zodiacText = open('zodiacDescriptions.txt')
#for line in zodiacText:
#   print(line)
```

Here's how we'll turn the contents of the file into a data type we can use:

```
#Load into a list

zodiacList = []
for line in zodiacText:
    zodiacList.append(line)

print(zodiacList)
```

We first create an empty list and then use a loop to move through the file line by line and add the line to the list. We do this using the *append* method of lists (note that it is a non-empty function). Of course we want to check that this has worked by printing the list.

4. Ask User for Birth Year and do the math

Users are notoriously problematic to deal with. Better to create a value to act as a placeholder for the moment and move on.

```
#Ask user for input (year)
birthYear = 1985
```

With a birth year in hand we can do the math and check the result. If this works then everything else is just a matter of fancy formatting and error trapping.

```
#Take year and use as a conditional (later we'll just access the dictionary
directly)

listLocation = (birthYear - 4) % 12
print(listLocation)
```

Why the "-4"? Our list is out of perfect alignment and fixing the math is easier than fixing the list. Be careful though, this is not always the case and it can be easy to forget.

We do get the right index value so we can move on.

5. Return the Zodiac Character

With the index value in hand we can directly return the character from our list:

```
#Return character
print("You are a ", zodiacList[listIndex])
```

6. *Really* Ask User for Input

Now that the core mechanics are working we can go back and actually collect input from the user. This is initially a one line change but in the long run it means we need to do error trapping.

```
#Ask user for input (year)
birthYear = raw_input("What year were you born: ")
listIndex = (birthYear - 4) % 12
print listIndex
```

7. Error Trapping

Try putting in anything that isn't a birth year and see what happens: ValueError. We can catch these with the following set up:

```
#Ask user for input (year)
try:
    birthYear=int(raw_input('What year were you born: '))
    listIndex = (birthYear - 4) % 12
    print listIndex

    #Return character
    print("You are a ", end="")
    print(zodiacList[listIndex])

except ValueError:
    print("You did not enter a number")
```

8. Repeat ==> Functions, loops, and conditionals

We very well may not make it this far. If that's the case then we'll at least walk through how to read this.

```
#Open the zodiac file

def ZodiacSetup():
    zodiacText = open('zodiacDescriptions2.txt')

    #Load into a list
    zodiacList = []
    for line in zodiacText:
        zodiacList.append(line)

    zodiacText.close()

    return zodiacList

def ZodiacFigure():
    #Ask user for input (year)
    try:
        birthYear=int(raw_input('What year were you born: '))
        #Take year and use as a conditional
        listIndex = (birthYear - 4) % 12
        print(listIndex)

        #Return character
        print("You are a ", end="")
        print(zodiacList[listIndex])

    except ValueError:
        print("You did not enter an integer")
        birthYear = "Stop"

    return birthYear

#Repeat
zodiacList = ZodiacSetup()

birthYear=0
while type(birthYear) is int:
    birthYear = ZodiacFigure()
```

9. Adding Graphics

This module still has to be built. It is unclear that there is enough time to do it within the workshop setting but it could be added for later reference or simply as a demonstration.