# Time to write some robust code

So far whatever code we have written, we have written free style coding. This means,

* We have not agreed to any standards or conventions to follow.
* We are not using any code template in developing automation script.
* We reported errors to console.
* We have not used modularity in our program. We followed top to bottom execution approach.
* We have not used exception handling.

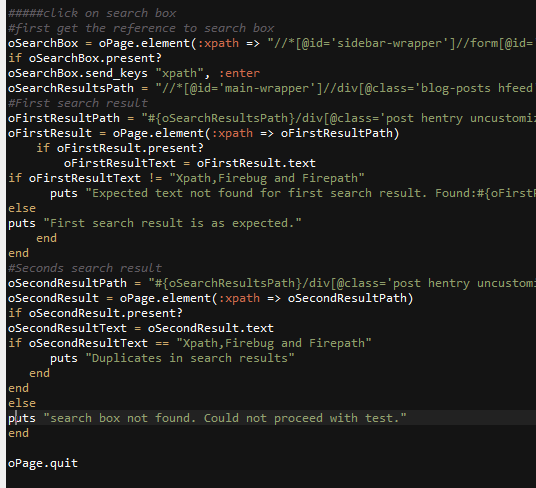
With such style of coding there are many drawbacks.

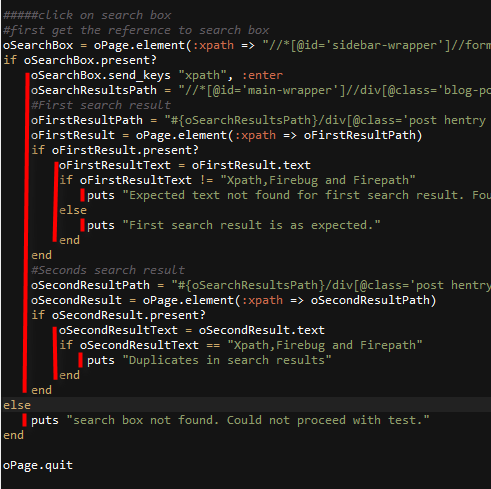
* Developing a framework to run a set of scripts wouldn’t be possible or easy.
* We will not have control over batch run.
* Production time increases.
* Maintenance cost increases, if automation code you have written is maintained by a different person.
* Managing an automation script even of an average size would be difficult.
* We don’t always run scripts in our PC. Sometimes we run our scripts on a remote machine or VM. In such cases error messages printed to console would not be available to end user located remotely. Even if they are available, under a batch run, understanding the messages printed on a console would be a night mare.
* To achieve a common functionality, we might have to repeat the code again and again. This would increase the size of your program.
* Any exception raised could terminate your automation run abruptly.

We can avoid such drawbacks at first place, and plan to write some robust code in following ways:

1. **Use comments to verbally explain what your script is trying to achieve.**
   * **This approach solves some of the above mentioned problems in following ways**
     1. Even a non-programmer would easily understand what the script is trying to do.
     2. Maintenance time reduces. One can quickly skim through comments to understand the code and concentrate on code where ever required.
     3. Increases program readability.
2. **Indent your code**
   * When a new block starts indent a level (add spaces to indicate start of a new block).
   * **This approach solves some of the above mentioned problems in following ways**
     1. Increases program readability.

Ex: Below is code without proper indentation.



Ex: Below is code with indentation.

1. **Don’t spam your logs with steps that are successfully executed. Avoid success statements.** 
   * If you post a message for successful execution of a step, your logs would get huge in size. This would consume time of users trying to analyze results. Users looking at the results would be interesting in knowing if something went wrong during execution but not about every successfully completed step. Just post warnings and errors.
2. **Introduce modularity in your code**
   * Make use of methods -
     1. To write code that would be used multiple times in your script. We can avoid code redundancy. Number of lines of code would be reduced significantly.
     2. To write an automation script. Place all you code in one method. This would give you the control on when to run automation. During batch run, you can dictate the order of execution of different test scripts.
   * Make use of Modules and Classes
     1. When the number of method count increases, group them into Classes and then Modules if necessary. Call them utilities and place in a separate file. For example, group methods that are common to all your web applications into a module/class called UtilsCommon and place it in a file called UtilsCommon.rb
     2. Make utility files as per the need. When automating a specific application, common operations of that specific application should go under a file.
   * **This approach solves some of the above mentioned problems in following ways**
     1. **Developing a framework would be easy.**

Ex: assume we are writing each test case in an individual file and assume we are writing all our code under one method (which need to be called explicitly for automation to run) whose name will be something like Run**ScriptName().**

Ex: if your script name is AutomateGoogle.rb then you would write automation code under Run**AutomateGoogle()**

In such scenario, by using methods and agreeing to a common convention of naming the method, we can design the framework to automatically call the method depending on selected script name from end user.

* + 1. **We will have control over batch run**

Assume we have following files and their respective automation code

1. AutomateGoogle.rb - Run**AutomateGoogle()**
2. AutomateGmail.rb - Run**AutomateGmail()**
3. AutomateYoutube.rb - Run**AutomateYoutube()**

Your batch file would be something like below. Observe how we can take control over what we want to execute. We dictate the order of methods execution, independent to the order of require

*#AutoTestList.rb – batch file name*

require 'AutomateGoogle.rb'

require 'AutomateGmail.rb'

require 'AutomateYoutube.rb'

*#############################################################*

*# Code to set up something. Initialization of workspace*

*#############################################################*

*#First execute automation for Gmail*

RunAutomateYoutube()

*#Second execute automation for Google*

RunAutomateGoogle()

*#Third execute automation for*

RunAutomateGmail()

*#############################################################*

*# Code to terminate process and clean workspace*

*#############################################################*

If methods are not used to write automation, your automation code will run in the order of files required. You will have no control.

* + 1. Production time decreases by making use of common utilities/methods that are written once and used multiple times.
    2. Maintenance cost reduces. Giving meaning full names to common methods and unities would reduce the user effort to understand what a group of lines will do.

Ex: UtilsCommon.ClickButton(Xpath) – Very much self-explanatory. Isn’t it?

* + 1. Size of a program reduces by making using of common methods.
    2. Many more…

1. **Use custom logger to log warnings, errors and messages**
2. **Use conventions to create variables, methods, utilities that team agreed to.**
3. **Use Exception handling technique where ever needed**
4. **Use custom template to write automation code**