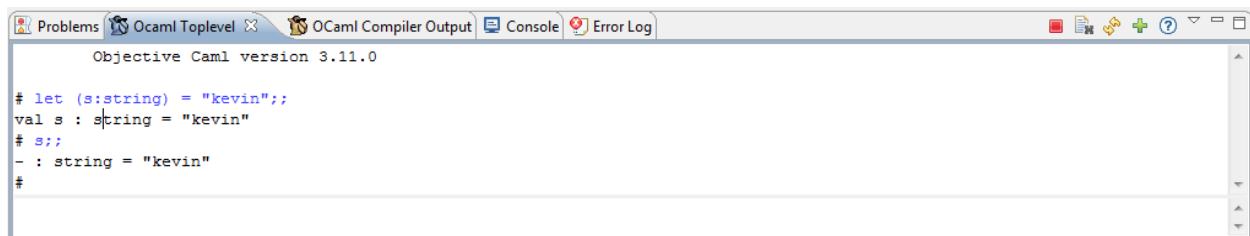


Using the OCaml Toplevel

The OCaml Toplevel is yet another useful feature that you can use to test your code. It's important to get to know how to use it in the coming weeks, since some of the homework will require it. For those of you who have worked in Dr. Java, the OCaml Toplevel functions similarly, but has a little bit more overhead. The OCaml Toplevel can be found on the bottom of the screen, and you bring the Toplevel to the front by clicking on the tab.

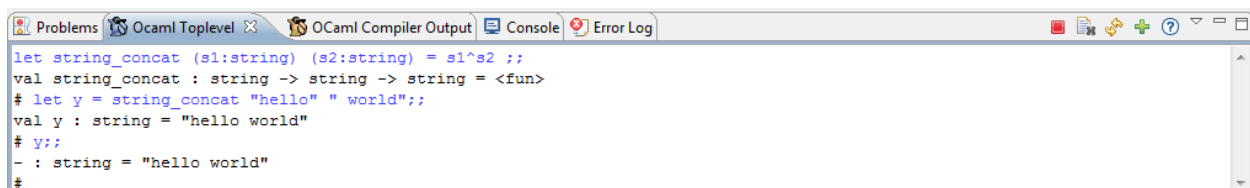
When you first start Toplevel session, it's a blank slate. All the other code in your folders, that you have open, etc. is all irrelevant. If you just want to test calculations and functions that you're going to write from scratch, Toplevel is an excellent way to do that.



```
Objective Caml version 3.11.0

# let (s:string) = "kevin";;
val s : string = "kevin"
# s;;
- : string = "kevin"
#
```

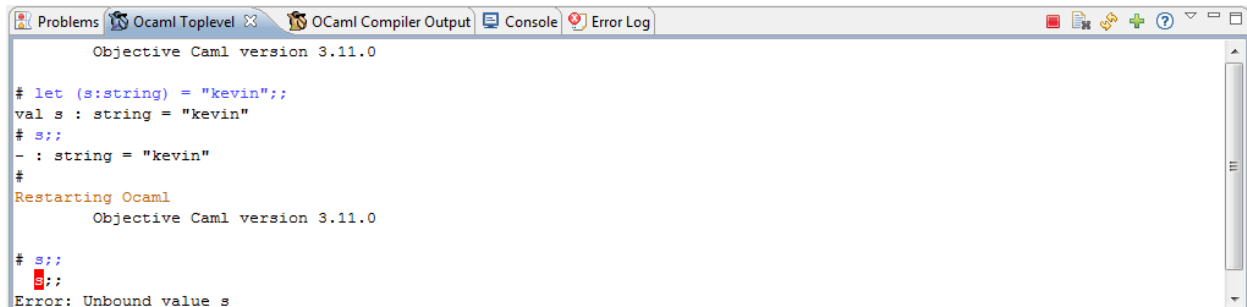
Let's first start by declaring a variable. In the example above, I first wrote the line 'let (s:string) = "kevin";;' The additional semicolons are needed to tell Toplevel that's the end of your input. If you want to check the value of a variable (in this case 's'), simple call it by typing its name "s;;" (followed with the two semicolons), and the Toplevel will print its value.



```
let string_concat (s1:string) (s2:string) = s1^s2 ;;
val string_concat : string -> string -> string = <fun>
# let y = string_concat "hello" " world";;
val y : string = "hello world"
# y;;
- : string = "hello world"
#
```

You can write functions in a similar way, just remember the semicolons at the end and to follow the conventions taught in class!

Let's say you want to reassign a variable or rewrite a method. But variables are immutable, so you can't do that. One way to restart is simply to hit the yellow refresh button which resets Toplevel to a clean slate. However keep in mind you'll lose everything you've typed into it before (that's why calling "s;;" again leads to an error)!



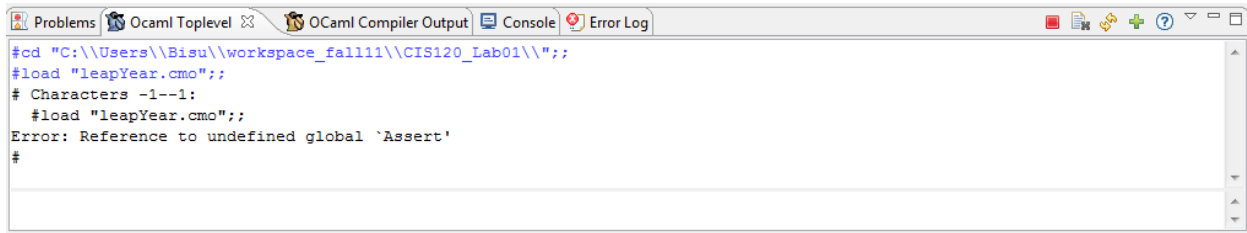
```
Objective Caml version 3.11.0

# let (s:string) = "kevin";;
val s : string = "kevin"
# s;;
- : string = "kevin"
#
Restarting Ocaml
Objective Caml version 3.11.0

# s;;
- : string = "kevin"
# s;;
Error: Unbound value s
```

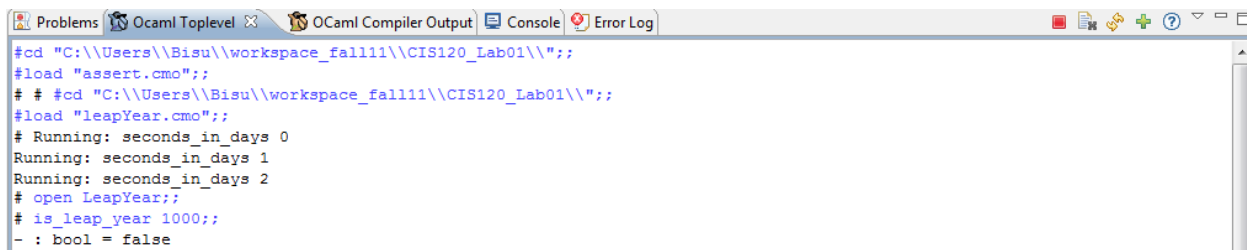
Now that you've have the basics down, we'll show you how useful Toplevel can be for testing code (and getting printouts of the values returned by functions). Using the Lab01 as an example, let's say you want to check just what values your functions are calculating. Test cases are great, but they only tell you if the result is right or wrong.

You want to test LeapYear.ml, however if you just load it into TopLevel (right clicking on the module and selecting "Load in Toplevel" will show an error:



```
#cd "C:\\Users\\Bisu\\workspace_fall111\\CIS120_Lab01\\";;
#load "leapYear.cmo";;
# Characters -1--1:
#load "leapYear.cmo";;
Error: Reference to undefined global `Assert'
#
```

The problem is that in order to use the functions you have to "load" AND "open" the module (and all modules that it requires). Looking at the error message, you'll realize that this module references the Assert module. So you'll have to load that first. After loading it, you have to make sure Toplevel is able to access it's commands, so make sure to type: "open LeapYear;;" (it's the module name with the first letter capitalized, and don't include the '.ml').



```
#cd "C:\\Users\\Bisu\\workspace_fall111\\CIS120_Lab01\\";;
#load "assert.cmo";;
# #cd "C:\\Users\\Bisu\\workspace_fall111\\CIS120_Lab01\\";;
#load "leapYear.cmo";;
# Running: seconds_in_days 0
Running: seconds_in_days 1
Running: seconds_in_days 2
# open LeapYear;;
# is_leap_year 1000;;
- : bool = false
```

Note: If you're really sharp you're probably wondering why you don't have to call open Assert. If you look on the top of the leapYear module, you'll see it already has called, open "Assert." Always remember to both load and open all your modules in the correct order!

If you've felt like you had enough, you can blast past here. For those that would like a bit more help with reading error messages in TopLevel, stick around for another page or so. Learn how to deal with errors could potentially save you a lot of time. Below I'll share some of the most common errors I've seen:

1) "Error: Reference to an undefined Global Mod..."

You've neglected to load a module! Keep in mind that most of the modules will require `Assert` (and others) before you can use them.

2) "Error: Unbound Value..."

Unfortunately this tends to be one of the most generic problems. One possibility is that you have typo (make sure your variable/function names are right). Another is that you forgot to open the module (with the functions you're calling).

3) "Error: This Expression has type <T1>, but used here with <T2>..."

You've probably run into this message in the workspace already. Here OCaml has found a typing error. Are you passing an `int` into where an `int list` should be? Are you trying to `cons` a list onto a list? Make sure that types match the function signatures or variable types.

Particularly for lists, tuples and more complicated data structures that you'll learn about later, `TopLevel` will be a powerful tool you have under your belt when your tests fail and have no idea why. Always feel free to ask if you have questions, otherwise Happy Coding! 8-)