CS1110: Thinking About Syntactic Errors

Python is a language and languages come equipped with rules of syntax. Python complains when you violate the rules. This handout is designed to help you interpret those complaints as they might typically arise here at the start of the course. We start with a *syntactically correct* Python program as it would appear in the Komodo editor:

```
 1   # Hyphenator2.py
 2   # the CS 1110 profs (cs-1110profs-L@cornell.edu)
 3   # Feb 2016
 4
 5   """ Inserts hyphens into a non-empty input string as follows:
 6   If the string has even length, the hyphen splits
 7   the first and second halves. Otherwise, a hyphen
 8   is inserted on either side of the middle character.
 9   """
10   s = input('Enter a string (remember to put quotes around it): ')
11   n = len(s)
12
13   if n%2 == 0:
14       # s has even length
15       m = n/2
16       h = s[0:m] + '-' + s[m:]
17   else:
18       #s has odd length
19       m = n/2
20       h = s[0:m]+'-'+s[m]+'-'+s[m+1:]
21
22   print s + ' becomes ' + h
```

The numbers at the left are line numbers and not part of the file. We are now going to induce a number of syntactic errors. For each induced error we will report "what Python says" and what it means in plain English.

Error 1. In Line 5 we change one of the `"""` to `""`.

Python says:

```
C:\Users\cv\Desktop>Hyphenator2
  File "C:\Users\cv\Desktop\Hyphenator2.py", line 5
    "" Inserts hyphens into a non-empty input string as follows:
              ^
SyntaxError: invalid syntax
```

Plain English Interpretation: Python is trying to decipher a line that begins with two double quotes and gets stuck with what follows. Any time Python points into what you think is a good doc string, check that you are using triple double quotes to delimit the start and finish of the doc string. Note that Python "points" with the little "carrot" symbol ^.

Error 2. In Line 17 remove the colon after the `else`.

Python says:

```
C:\Users\cv\Desktop>Hyphenator2
  File "C:\Users\cv\Desktop\Hyphenator2.py", line 17
    else
       ^
SyntaxError: invalid syntax
```

Plain English Interpretation: If you leave off a colon in an `if-else` construction, then Python will point to the site of your forgetfulness.

Error 3. In Line 13 change `n%2 == 0` to `n%2 = 0`

Python says:

```
C:\Users\cv\Desktop>Hyphenator2
  File "C:\Users\cv\Desktop\Hyphenator2.py", line 13
    if n%2 = 0:
           ^
SyntaxError: invalid syntax
```

Plain English Interpretation: A single `=` is used in the assignment statement. In Boolean expressions the only allowable comparison operations are `==`, `<=` , `<` , `>=`, `>` and `!=`. If you use something besides these in a Boolean expression, then Python will point to the offending comparison.

Error 4. In Line 13 change `n%2 == 0` to `n%2 = = 0`.

Python says:

```
C:\Users\cv\Desktop>Hyphenator2
  File "C:\Users\cv\Desktop\Hyphenator2.py", line 13
    if n%2 = = 0:
           ^
SyntaxError: invalid syntax
```

Plain English Interpretation: Again, Python is complaining about an illegal comparison operation. If you are testing for equal values, you use `==` with no blanks in between the first `=` and the second `=`.

Error 5. In Line 10, remove one of the quotes in the `input` statement.

Python says:

```
C:\Users\cv\Desktop>Hyphenator2
  File "C:\Users\cv\Desktop\Hyphenator2.py", line 10
    s = input(Enter a string (remember to put quotes around it): ')
                    ^
SyntaxError: invalid syntax
```

Plain English Interpretation: Without the lead quote Python thinks that `Enter a` names two variables and it is telling you that you cannot write two variables side by side like that in this context. Python doesn't say "you forgot the single quote" because to its way of thinking, perhaps you meant `Enter + a` or `Enter, a`. Moral: if Python points to what you think is a perfectly good string, check that the string is properly delimited with a single quote at its start and at its end.

Error 6. In Line 3, remove the `"#"`.

Python says:

```
C:\Users\cv\Desktop>Hyphenator2
  File "C:\Users\cv\Desktop\Hyphenator2.py", line 3
    Feb 2016
    ^
IndentationError: unexpected indent
```

Plain English Interpretation: Python is trying to decipher `Feb 2016` as a meaningful statement and notices that it isn't tabbed properly. Moral. If Python points to what you think is a perfectly good comment, make sure it begins with a `#`.

Error 7. In Line 19, un-indent `m = n/2`.

Python says:

```
C:\Users\cv\Desktop>Hyphenator2
  File "C:\Users\cv\Desktop\Hyphenator2.py", line 19
    m = n/2
    ^
IndentationError: expected an indented block
```

Plain English Interpretation: Un-indenting line 19 confuses Python because line 20 remains indented. Indentation after the `if` is how Python figures out what to do if the Boolean expression is `True`. The Komodo editor generally handles indentation for you. But you can (accidentally) override it. Always look at the indentation in your code and make sure it signals the right thing to Python.