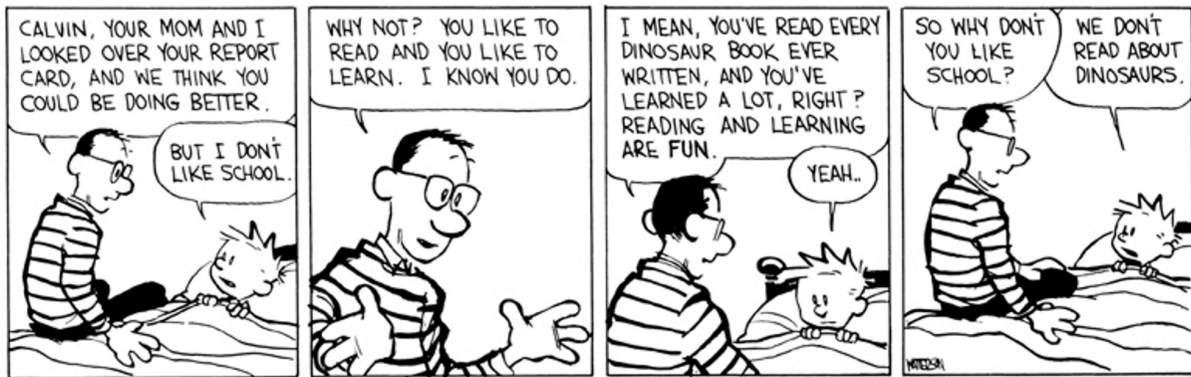


CSCI 134: quiz 1 practice 2

This is a practice quiz for quiz 1. The length and difficulty of the real quiz will be about the same level or easier :)

0. Boolean logic



OCTOBER 16, 17, 18, 1989

Implement Calvin's logic for when he would like to go to school. The function should return True when his conditions are met, and False otherwise. If you use anything other than boolean operators (and/or/not) and return, you're probably overthinking it.

```
def school_is_good(must_read, must_learn, num_dinosaurs):
```

```
    # must_read is a boolean variable
```

```
    # must_learn is a boolean variable
```

```
    # num_dinosaurs is an integer
```

1. reading conditionals, loops, scope, and slicing

What do the following snippets of code print? Error is a valid option

1a

```
singer = "michael jackson"
```

```
def pop_music(singer):  
    output = ""  
    if singer == "michael jackson":  
        output = output + "king of pop"  
    output = output + " " + singer  
    print(output)
```

```
def canadian_pop_music(singer):  
    output = ""  
    if singer == "michael jackson":  
        output = "replacing " + singer + " with"  
        singer = "michael buble"  
    output = output + " " + singer  
    print(output)
```

```
pop_music(singer)  
canadian_pop_music(singer)  
print(singer)
```

1b

```
x = 5
```

```
print(x == 4 or 3)
```

1c

```
def decode(message):
```

```
    message = message[0:len(message)-3]
```

```
    decoded_message = []
```

```
    curr_word = ""
```

```
    for letter in message :
```

```
        if letter == "_" and curr_word[0] == "s" :
```

```
            decoded_message = decoded_message + [curr_word]
```

```
            curr_word = ""
```

```
        elif letter == "_" :
```

```
            curr_word = ""
```

```
        else:
```

```
            curr_word = curr_word + letter
```

```
    return decoded_message
```

```
print(decode("feisty_foxes_silly_squirrels_fainting_snatching_snacks_set"))
```

2. writing conditionals (if-elif-else)

Write a function that returns a boolean expression corresponding to whether a 3 digit number is a “sandwich” number. A number is a sandwich number if

- The first and last digits of the number are both 1, or
- The first and last digits of the number are equal to each other and also even

Any other 3 digit number is not a sandwich number

E.g., 151 and 262 are sandwich numbers, but 363 is not.

```
def is_sandwich_number(number):
```

```
    # implement your logic (can assume number is a 3 digit positive integer)
```

3. writing loops and using functions

Write a function that uses the `is_sandwich_number` from part 2 (you can assume it is correctly implemented) to return a subset of numbers from a given list of 3 digit numbers that correspond to sandwich numbers.

E.g.,

- The input `[312, 454, 181, 777, 888, 999]` should return `[454, 181, 888]`

```
def compute_sandwich_sublist(list_of_numbers):
```

```
    # implement your logic (can assume every number in the list is 3 digits)
```