Python Problem Set - 1

1. WAP to find factorial of a number
2. WAP to find largest among three numbers
3. WAP to check whether it's a leap year or not
4. WAP to check Armstrong number
5. WAP to count vowels in a string
6. WAP to print odd numbers from the input except 3 and 15
7. WAP to display all natural numbers up to 100 which are not divisible by 5
8. Define a function that computes the length of a given list or string without using len().
9. The web is built with HTML strings like "<i>Yay</i>" which draws Yay as italic text. In this example, the "i" tag makes <i> and </i> which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "<i>Yay</i>".
    - make_tags('i', 'Yay') → '<i>Yay</i>'
    - make_tags('i', 'Hello') → '<i>Hello</i>'

10. Given a string, return the string made of its first two chars, so the String "Hello" yields "He". If the string is shorter than length 2, return whatever there is, so "X" yields "X", and the empty string "" yields the empty string "".
11. Given 2 strings, return their concatenation, except omit the first char of each. The strings will be at least length 1.
    - non_start('Hello', 'There') → 'ellohere'
    - non_start('java', 'code') → 'avaode'
12. Given a string, return a string where for every char in the original, there are two chars.
    - double_char('The') → 'TThhee'
    - double_char('AAbb') → 'AAAAbbbb'
    - double_char('Hi-There') → 'HHii--TThheerree'
13. Return the number of times that the string "hi" appears anywhere in the given string.
    - count_hi('abc hi ho') → 1
    - count_hi('ABChi hi') → 2
    - count_hi('hihi') → 2
14. Given two strings, return True if either of the strings appears at the very end of the other string, ignoring upper/lower case differences (in other words, the computation should not be "case sensitive"). Note: s.lower() returns the lowercase version of a string.

- end_other('Hiabc', 'abc') → True
- end_other('AbC', 'HiaBc') → True
- end_other('abc', 'abXabc') → True

15. Given 3 int values, a b c, return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.
    - lucky_sum(1, 2, 3) → 6
    - lucky_sum(1, 2, 13) → 3
    - lucky_sum(1, 13, 3) → 1
16. Given three ints, a b c, return True if one of b or c is "close" (differing from a by at most 1), while the other is "far", differing from both other values by 2 or more. Note: abs(num) computes the absolute value of a number.
    - close_far(1, 2, 10) → True
    - close_far(1, 2, 3) → False
    - close_far(4, 1, 3) → True
17. Given 2 ints, a and b, return their sum. However, sums in the range 10..19 inclusive, are forbidden, so in that case just return 20.
    - sorta_sum(3, 4) → 7
    - sorta_sum(9, 4) → 20
    - sorta_sum(10, 11) → 21
18. Given a day of the week encoded as 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating if we are on vacation, return a string of the form "7:00" indicating when the alarm clock should ring. Weekdays, the alarm should be "7:00" and on the weekend it should be "10:00". Unless we are on vacation -- then on weekdays it should be "10:00" and weekends it should be "off".
    - alarm_clock(1, False) → '7:00'
    - alarm_clock(5, False) → '7:00'
    - alarm_clock(0, False) → '10:00'
19. Given a number n, return True if n is in the range 1..10, inclusive. Unless "outsideMode" is True, in which case return True if the number is less or equal to 1, or greater or equal to 10.
    - in1to10(5, False) → True
    - in1to10(11, False) → False
    - in1to10(11, True) → True

20. Given a list of integers, return the difference between the largest and smallest values in the list.
21. Given a list of integers, return True if the list contains 2 consecutive occurrences of 3.
22. Given a list of Fruits, remove the 3rd and the 5th element from the list.
23. WAP that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates. Use functions.
24. Define a function histogram() that takes a list of integers and prints a histogram to the screen. For example, histogram([4, 9, 7]) should print the following:

```
****

*********

*******
```

25. Create a new dictionary called prices using {} format like the example above.
    - Put these values in your prices dictionary:

      ```
      "banana": 4,

      "apple": 2,

      "orange": 1.5,

      "pear": 3
      ```

    - Loop through each key in prices. For each key, print out the key along with its price and stock information. Print the answer in the following format:

      ```
      apple

      price: 2

      stock: 0
      ```

    - Let's determine how much money you would make if you sold all of your food.
    - Create a variable called total and set it to zero.
    - Loop through the prices dictionaries. For each key in prices, multiply the number in prices by the number in stock. Print that value into the console and then add it to total.
    - Finally, outside your loop, print total.
26. First, make a list called groceries with the values "banana", "orange", and "apple".
    - Define this two dictionaries:

      ```
      stock = {

      "banana": 6,

      "apple": 0,

      "orange": 32,

      "pear": 15

      }
      ```

    - prices = {

```
"banana": 4,

"apple": 2,

"orange": 1.5,

"pear": 3

}
```

- Define a function compute_bill that takes one argument food as input. In the function, create a variable total with an initial value of zero. For each item in the food list, add the price of that item to total. Finally, return the total. Ignore whether or not the item you're billing for is in stock. Note that your function should work for any food list.
- Make the following changes to your compute_bill function:
- While you loop through each item of food, only add the price of the item to total if the item's stock count is greater than zero.
- If the item is in stock and after you add the price to the total, subtract one from the item's stock count.

27. Represent a small bilingual lexicon as a Python dictionary in the following fashion {"merry":"god", "christmas":"jul", "and":"och", "happy":gott", "new":"nytt", "year":"år"} and use it to translate your Christmas cards from English into Swedish. That is, write a function translate() that takes a list of English words and returns a list of Swedish words.