Better programmer tips

1. Proper naming should be given to variables

2. Proper spacing

Class & Object

```
class Student:
  #class Attribute
  college = "MREC"
  #constructor
  def __init__(self, name, branch, roll_no):
    #Instance Attributes
    self.name = name
    self.branch = branch
    self.roll_no = roll_no

  #Instance method
  def student_data(self):
    return [Student.college, self.name, self.branch, self.roll_

#driver code
#Object creation
obj1 = Student("sai", "CSE", "14j41A0525")
print( obj1.student_data() )

```

```
['MREC', 'sai', 'CSE', '14j41A0525']
```

Anagram

Q1. Anagram

Problem

Given two strings, check to see if they are anagrams.

An anagram is when the two strings can be written using the exact same letters (so you can just rearrange the letters to get a different phrase or word).

For example:

"public relations" is an anagram of "crap built on lies."

"clint eastwood" is an anagram of "old west action"

Note: Ignore spaces and capitalization.

So "d go" is an anagram of "God" and "dog" and "o d g".

```
class Solution:
  def is_anagrams(self, sentence_1, sentence_2):
    sentence_1 = sentence_1.replace(" ", "").lower()
    sentence_2 = sentence_2.replace(" ", "").lower()
    return sorted(sentence_1) == sorted(sentence_2)
#driver code
sentence_1 = input("Enter the string 1: ")
sentence_2 = input("Enter the string 2: ")
#object creation
obj = Solution()
print( obj.is_anagrams(sentence_1, sentence_2))
```

```
Enter the string 1: client eastwood
Enter the string 2: old west action
False
```

Sentence reversal

Q2. Sentence Reversal

Problem

Given a string of words, reverse all the words.

For example:

Given:

'This is the best'

Return:

'best the is This'

```
class Solution:
  def reverse_words(self, sentence):
    result = ""
    words_list = sentence.split(" ")
    for word in words_list[::-1]:
      result = result + word + " "
    return result
#driver code
sentence = input("Enter the sentence: ")
obj = Solution()
print( obj.reverse_words(sentence) )
```

```
Enter the sentence: This is the best
best the is This
```

```
#Solution 2
class Solution:
  def reverse_words(self, sentence):
    words_list = sentence.split(" ")
    return " ".join(words_list[::-1])
#driver code
sentence = input("Enter the sentence: ")
obj = Solution()
print( obj.reverse_words(sentence) )
```

```
Enter the sentence: This is the best
best the is This
```

String compression

Q3. String Compression
Problem
Given a string in the form 'AAAABBBBCCCCCDDEEEE' compress it to become 'A4B4C5D2E4'. For this problem, you can falsely "compress" strings of single or double letters. For instance, it is okay for 'AAB' to return 'A2B1' even though this technically takes more space.
The function should also be case sensitive,
so that a string 'AAAaaa' returns 'A3a3'.

SI: AAAaaab
SO: A3a3b1

```
class Solution:
  def character_count(word):
    unique = ""
    result = ""
    for character in word:
      if character not in unique:
        unique += character
        result += character + str(word.count(character))
    return result
#driver code
word = input("Enter the word: ")
obj = Solution
print( obj.character_count(word) )
```

```
Enter the word: aaaaDDDgggh
a4D3g3h1
```

```
#set is unordered and store only unique values
unique = set("AAAaaab")
print(unique)
```

```
{'a', 'A', 'b'}
```

Comparing strings without using inbuilt functions

Q4. Take in Two Strings and Display the Larger String without Using Built-in Functions

Sample Input:

Delhi

Bangalore

Sample Output:

Bangalore

```
class Solution:
  def compare_strings(self, sentence_1, sentence_2):
    count_1, count_2 = 0, 0
    for character in sentence_1:
      count_1 += 1
    for character in sentence_2:
      count_2 += 1
    if(count_1 > count_2):
      return sentence_1
    elif(count_1 < count_2):
      return sentence_2
    else:
      return "Both are having equal length"

#driver code
sentence_1 = input("Enter the sentence 1: ")
sentence_2 = input("Enter the sentence 2: ")
obj = Solution()
print(obj.compare_strings(sentence_1, sentence_2))
```

```
Enter the sentence 1: mrec
Enter the sentence 2: data science
data science
```

Max count of words

Q5. A sentence is a list of words that are separated by a single space with no leading or trailing spaces. You are given an array of strings sentences, where each sentences[i] represents a single sentence. Return the maximum number of words that appear in a single sentence.

Example :

Input:

sentences = ["alice and bob love leetcode", "i think so too", "this is great thanks very much"]

Output: 6

```
class Solution:
  def max_words(self, sentences):
    max_count = 0
    for sentence in sentences:
```

```
    #Removing begining and ending spaces
    sentence = sentence.strip()
    words_count = sentence.count(" ") + 1
    if(words_count > max_count):
      max_count = words_count
  return max_count
#driver code
sentences = list(input().split(","))
obj = Solution()
print( obj.max_words(sentences) )
```

```
mrec hyderabad, mining hyd india, data science hyd india world
5
```

numeric words to integer

Q6. Given a string S, containing numeric words, the task is to convert the given string to the actual number.

Example:
Input: S = “four zero one four”
Output: 4014

```
class Solution:
  def words_to_int(self, num_words):
    result = ""
    digits = {"zero" : 0, "one" : 1, "two" : 2, "three" : 3,
              "four" : 4, "five" : 5, "six" : 6, "seven" : 7,
              "eight" : 8, "nine" : 9}
    words_list = num_words.split(" ");
    for word in words_list:
      result += str( digits[word] )
    return int(result)


#driver code
num_words = input("Enter the numeric words: ")
obj = Solution()
print( obj.words_to_int(num_words) )
```

```
Enter the numeric wordsfour eight two three
4823
```

Inheritance

```
#Inheritance Example
class Human:     #parent class
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender
    def description(self):
        print(f"Hey! My name is {self.name}, I'm a {self.gender

class Boy(Human):    #child class
    def schoolName(self, schoolname):
        print(f"I study in {schoolname}")

#driver code
b = Boy('John', 15, 'male')
b.description()
b.schoolName("Sunshine Model School")
```

```
Hey! My name is John, I'm a male and I'm 15 years old
I study in Sunshine Model School
```

Polymorphism

1. Compile time polymorphism

```
#method overloading
class Example:
    def add(self, a, b = 0, c = 0):
        return a + b + c

obj = Example()
print(obj.add(1))       # Output: 1
print(obj.add(1, 2))     # Output: 3
print(obj.add(1, 2, 3))  # Output: 6
```

```
1
3
6
```

Run time polymorphism

```
#Method overriding
class Animal:
    def make_sound(self):
        print("Some generic sound")

class Dog(Animal):
    def make_sound(self):
        print("Bark! Bark!")

class Cat(Animal):
    def make_sound(self):
        print("Meow!")

# Create instances of the subclasses
dog = Dog()
cat = Cat()

# Call the overridden method
dog.make_sound()  # Output: Bark! Bark!
cat.make_sound()  # Output: Meow!


```

```
Bark! Bark!
Meow!
```

X pattern

```
def x_pattern(lines):
  for i in range(0, lines):
    for j in range(0, lines):
      if(i == j or i + j == lines - 1):
        print("*", end = "")
      else:
        print(" ", end = "")
    print()

#driver code
lines = int(input("Enter the no of lines: "))
x_pattern(lines)
```

```
Enter the no of lines: 5
*   *
 * *
  *
 * *
*   *
```

Alphabet X Pattern

```
def alphabet_pattern(lines):
  character = 'A'
  for i in range(0, lines):
    for j in range(0, lines):
      if(i == j or i + j == lines - 1):
        print(character, end = "")
        character = chr( ord( character ) + 1 )
      else:
        print(" ", end = "")
    print()

#driver code
```

```
lines = int(input("Enter the no of lines: "))
alphabet_pattern(lines)
```

```
Enter the no of lines: 5
A   B
```