Better programmer tips

1. Proper naming should be given to variables

2. Proper spacing

Encapsulation

public

```
# illustrating public members & public access modifier
class pub_mod:
    # constructor
    def __init__(self, name, age):
        self.name = name;
        self.age = age;

    def Age(self):
        # accessing public data member
        print("Age: ", self.age)
# creating object
obj = pub_mod("Jason", 35);
# accessing public data member
print("Name: ", obj.name)
# calling public member function of the class
obj.Age()

```

```
Name:  Jason
Age:  35
```

Private

```
# illustrating private members & private access modifier
class Rectangle:
  __length = 0 #private variable
  __breadth = 0 #private variable
  def __init__(self):
    #constructor
    self.__length = 5
    self.__breadth = 3
    #printing values of the private variable within the class
    print(self.__length)
    print(self.__breadth)

rect = Rectangle() #object created
#printing values of the private variable outside the class
print(rect.__length)
print(rect.__breadth)

```

```
5
3
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-2-8c79836a2811> in <cell line: 15>()
     13 rect = Rectangle() #object created
     14 #printing values of the private variable outside the class
---> 15 print(rect.__length)
     16 print(rect.__breadth)

AttributeError: 'Rectangle' object has no attribute '__length'
```

Public methods

```
#Public method to access private members
class MyClass:
    def __init__(self, private_member):
        self.__private_member = private_member

    def get_private_member(self):
        return self.__private_member

    def set_private_member(self, new_value):
        self.__private_member = new_value

# Driver code
obj = MyClass(42)
value = obj.get_private_member()
print(value)
obj.set_private_member(100)
new_value = obj.get_private_member()
print(new_value)


```

```
42
100
```

Name Mangling

```
class MyClass:
    def __init__(self):
        self.__private_member = 42

obj = MyClass()
# Name mangling to access private member
print(obj._MyClass__private_member)
```

```
42
```

Protected

```
# illustrating protected members & protected access modifier
class details:
    _name="Jason"
    _age=35
    _job="Developer"
class pro_mod(details):
    def __init__(self):
        print(self._name)
        print(self._age)
        print(self._job)

# creating object of the class
obj = pro_mod()
```

Abstraction

```
#Abstraction example
from abc import ABC, abstractmethod
class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return 3.14 * self.radius * self.radius

class Square(Shape):
    def __init__(self, side):
        self.side = side
    def area(self):
        return self.side * self.side
#Driver code
obj1 = Circle(2)
obj2 = Square(5)
print(obj1.area())
print(obj2.area())
```

```
12.56
25
```

Regular Expressions

```
print("sai\nmrec")
print(r"sai\nmrec")
#print("sai\\nmrec")
```

```
sai
mrec
sai\nmrec
```

Double-click (or enter) to edit

```
#search( )
import re
s = 'computer science portal'
match = re.search(r'portal', s)
print('Start Index:', match.start())
print('End Index:', match.end())

```

```
Start Index: 17
End Index: 23
```

```
#findall( )
import re
txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

```
['ai', 'ai']
```

Q1. first pallindrome in a list of strings

Question 1:

Given an array of strings words, return the first palindromic string in the array. If there is no such string, return an empty string "".

A string is palindromic if it reads the same forward and backward.

Input: words = ["abc","car","ada","racecar","cool"]

Output: "ada"

```
class Solution:
  def first_pallindrome(self, strings_list):
    for word in strings list:
```

```
    if(word == word[::-1]):
      return word
  return ""
#driver code
strings_list = list(input().split(","))
obj = Solution()
print(obj. first_pallindrome(strings_list))
```

```
abc,car,ada,racecar,cool
ada
```

```
def for_loop():
  for i in range(1, 11):
    if(i == 3):
      print(i)
      return
  print("I am outside for loop")
#driver code
for_loop()
```

```
3
```

Q2. Opposite sign digits sum

Question 2:

You are given a positive integer n. Each digit of n has a sign according to the following rules:

The most significant digit is assigned a positive sign.
Each other digit has an opposite sign to its adjacent digits.
Return the sum of all digits with their corresponding sign.

Input: n = 886996
Output: 0
Explanation: (+8) + (-8) + (+6) + (-9) + (+9) + (-6) = 0.

```
class Solution:
  def opp_sign_sum(self, num):
    num_str = str(num)
    total_sum = 0
    for i in range(0, len(num_str)):
      if(i % 2 == 0):
        total_sum += int(num_str[i])
      else:
        total_sum -= int(num_str[i])
    return total_sum

#driver code
num = int(input())
obj = Solution()
print(obj.opp_sign_sum(num))
```

```
886996
0
```

```
#Another way
class Solution:
  def opp_sign_sum(self, num):
    num_str = str(num)
    total_sum = 0
    sign = 1
    for digit in num_str:
      digit = int(digit) * sign
      total_sum += digit
      sign = sign * (-1)
    return total_sum

#driver code
num = int(input())
obj = Solution()
print(obj.opp_sign_sum(num))

```

```
881
1
```

```
def for_loop():
  for i in range(1, 11):
    if(i == 3):
      print(i)
      break
  print("I am outside for loop")
#driver code
for_loop()
```

```
3
I am outside for loop
```

Triplet

Question 3:

You are given a 0-indexed, strictly increasing integer array nums and a positive integer diff. A triplet (i, j, k) is an arithmetic triplet if the following conditions are met:

$i < j < k$,

nums[j] - nums[i] == diff, and

nums[k] - nums[j] == diff.

Return the number of unique arithmetic triplets.

Input: nums = [0,1,4,6,7,10], diff = 3

Output: 2

Explanation:

(1, 2, 4) is an arithmetic triplet because both 7 - 4 == 3 and 4 - 1 == 3.

(2, 4, 5) is an arithmetic triplet because both 10 - 7 == 3 and 7 - 4 == 3.

```
class Solution:
  def arithmetic_triplet(self, nums, diff):
    count = 0
    for i in range(0, len(nums)):
      for j in range(i + 1, len(nums)):
        for k in range(j + 1, len(nums)):
          if(nums[j] - nums[i] == diff and nums[k] - nums[j] ==
            count += 1
    return count
#driver code
nums = list(map(int, input().split(" ")))
diff = int(input())
obj = Solution()
print(obj.arithmetic_triplet(nums, diff))
```

```
0 1 4 6 7 10
3
2
```

absolute difference count

Question 4:

Given an integer array nums and an integer k, return the number of pairs (i, j) where i < j such that |nums[i] - nums[j]| == k.

The value of |x| is defined as:

x if x >= 0.

-x if x < 0.

Input: nums = [1,2,2,1], k = 1

Output: 4

Explanation: The pairs with an absolute difference of 1 are:

- [1,2,2,1]
- [1,2,2,1]
- [1,2,2,1]
- [1,2,2,1]

```
class Solution:
  def abs_diff_pairs(self, nums, diff):
    count = 0
    for i in range(0, len(nums)):
      for j in range(i + 1, len(nums)):
        if(abs(nums[i] - nums[j]) == diff):
          count += 1
    return count
#driver code
nums = list(map(int, input().split(" ")))
diff = int(input())
obj = Solution()
print(obj.abs_diff_pairs(nums, diff))
```

```
1 2 2 1
1
4
```

Sorted array target index

Question 5:

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

Example 1:

Input: nums = [1,3,5,6], target = 5

Output: 2

Example 2:

Input: nums = [1,3,5,6], target = 7

Output: 4

```
class Solution:
  def target_index(self, nums, target):
    if target in nums:
      return nums.index(target)
```

```
    else:
      nums.append(target)
      nums.sort()
      return nums.index(target)
#driver code
nums = list(map(int, input().split(' ')))
target = int(input())
obj = Solution()
print(obj.target_index(nums, target))
```

```
1 5 9 11 45 89 100
92
6
```

Topics covered

Python Introduction

Data types

Keywords

Conditional Statements

Loops

List

Slicing

Strings

Tuples

Dictionary

Sets

Differences b/w List, Tuple, Dictionary, Set

Functions, Recursion

Exceptional Handling

Modules

Math module

Datetime module

File Handling

OOPs intoduction

Class

Object

Attributes

OOPs concepts

Inheritance

Polymorphism

Encapsulation

Abstraction

Regular expressions

re module

Python database communication

sqllite3

40 Programs

String pattern

```
class Solution:
  def string_pattern(self, word):
    for i in range(0, len(word)):
      for j in range(0, len(word)):
        if(i == j):
          print(word[i], end = "")
        else:
          print(" ", end = "")
      print()
#driver code
word = input()
obj = Solution()
obj.string_pattern(word)
```

```
data
d
 a
```

```
t
 a
```

Square Pattern

```
class Solution:
  def square_pattern(self, num):
    for i in range(0, num):
      for j in range(0, num):
        if(i == 0 or j == 0 or i == num - 1 or j == num - 1):
          print("*", end = "")
        else:
          print(" ", end = "")
      print()
#driver code
num = int(input())
obj = Solution()
obj.square_pattern(num)
```

```
5
```