Better programmer tips

1. Proper naming should be given to variables

2. Proper spacing

Differences.png

Dictionary continuation...

```
dict_values = {0 : "Apple", 1 : "Hard", 2 : "Work", "new_set" :
print(dict_values.get("new_set"))
```

```
(1, 5, 8)
```

```
print(dict_values["new_set"])
```

```
(1, 5, 8)
```

```
print(dict_values.pop("new_set"))
```

```
(1, 5, 8)
```

```
del dict_values[2]
print(dict_values)
```

```
{0: 'Apple', 1: 'Hard'}
```

```
print(dict_values.popitem())
```

```
(1, 'Hard')
```

```
print(dict_values)
```

```
{0: 'Apple'}
```

Set

```
#Creating an empty set
set_values = set()
print(type(set_values))
```

```
<class 'set'>
```

```
#set
set_values = {2, "python", 'a', 5, 9.4}
print(set_values[0])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-b64524a4120f> in <cell line: 3>()
      1 #set
      2 set_values = {2, "python", 'a', 5, 9.4}
----> 3 print(set_values[0])

TypeError: 'set' object is not subscriptable
```

```
#set - printing values in different lines
set_values = {2, "python", 'a', 5, 9.4}
for value in set_values:
  print(value)
```

```
2
5
python
a
9.4
```

```
#set - printing values in the same line
set_values = {2, "python", 'a', 5, 9.4}
for value in set_values:
  print(value, end = " ")
```

```
2 5 python a 9.4
```

```
#in operator
set_values = {2, "python", 'a', 5, 9.4}
print(7 in set_values)
print("python" in set_values)
```

```
False
True
```

```
#add function
set_values = {2, "python", 'a', 5, 9.4}
set_values.add("apple")
print(set_values)
```

```
{2, 5, 'python', 'a', 9.4, 'apple'}
```

```
#len function
set_values = {2, "python", 'a', 5, 9.4}
print(len(set_values))
```

```
5
```

```
#remove function
set_values = {2, "python", 'a', 5, 9.4}
set_values.remove("python")
print(set_values)
```

```
{2, 5, 'a', 9.4}
```

```
#discard function
set_values = {2, "python", 'a', 5, 9.4}
set_values.discard("python")
print(set_values)
```

```
{2, 5, 'a', 9.4}
```

```
#discard and remove difference
set_values = {2, "python", 'a', 5, 9.4}
set_values.discard("apple")
print(set_values)
```

```
{2, 5, 'python', 'a', 9.4}
```

```python
#discard and remove difference
set_values = {2, "python", 'a', 5, 9.4}
set_values.remove("apple")
print(set_values)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-29-568cddefc7ba> in <cell line: 3>()
      1 #discard and remove difference
      2 set_values = {2, "python", 'a', 5, 9.4}
----> 3 set_values.remove("apple")
      4 print(set_values)

KeyError: 'apple'
```

```python
#clear()
set_values = {2, "python", 'a', 5, 9.4}
set_values.clear()
print(set_values)
```

```
set()
```

Double-click (or enter) to edit

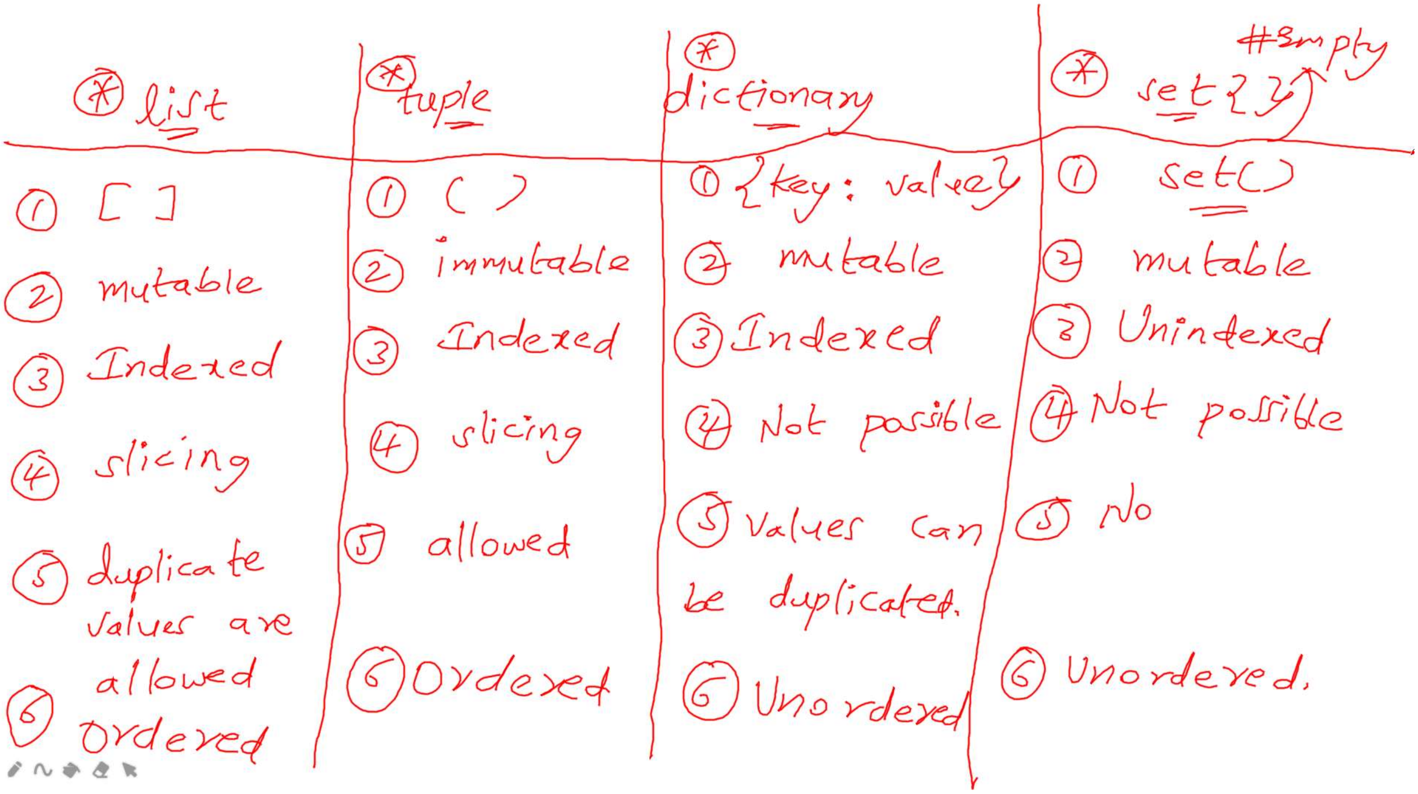
```python
#union
month1_set = {"Jan", "Feb", "Dec", "Mar"}
month2_set = {"May", "Jan", "Oct", "Mar"}
print(month1_set | month2_set)
print(month1_set.union(month2_set))
```

```
{'Mar', 'Dec', 'Oct', 'Jan', 'Feb', 'May'}
{'Mar', 'Dec', 'Oct', 'Jan', 'Feb', 'May'}
```

```python
#intersection
month1_set = {"Jan", "Feb", "Dec", "Mar"}
month2_set = {"May", "Jan", "Oct", "Mar"}
print(month1_set & month2_set)
print(month1_set.intersection(month2_set))
```

```
{'Jan', 'Mar'}
{'Jan', 'Mar'}
```

Differences



* list	* tuple	* dictionary	* set{} #empty
① [ ]	① ( )	① {key: value}	① set()
② mutable	② immutable	② mutable	② mutable
③ Indexed	③ Indexed	③ Indexed	③ Unindexed
④ slicing	④ slicing	④ Not possible	④ Not possible
⑤ duplicate values are allowed	⑤ allowed	⑤ Values can be duplicated.	⑤ No
⑥ Ordered	⑥ Ordered	⑥ Unordered	⑥ Unordered.

Functions

Q1. Problem Statement :

You have to write a function that accepts a string of length “length”, the string has some “#”, in it. you have to move all the hashes to the front of the string and return the whole string back and print it.

Example:
Sample Test Case
Input:
Move#Hash#to#Front
Output:
###MoveHashtoFront

```
def move_hash(words):
  hash_count = words.count("#")
  words = words.replace("#", "")
  words = hash_count * "#" + words
  return words

#driver code
words = input()
print( move_hash(words) )
```

```
Move#Hash#to#Front
###MoveHashtoFront
```

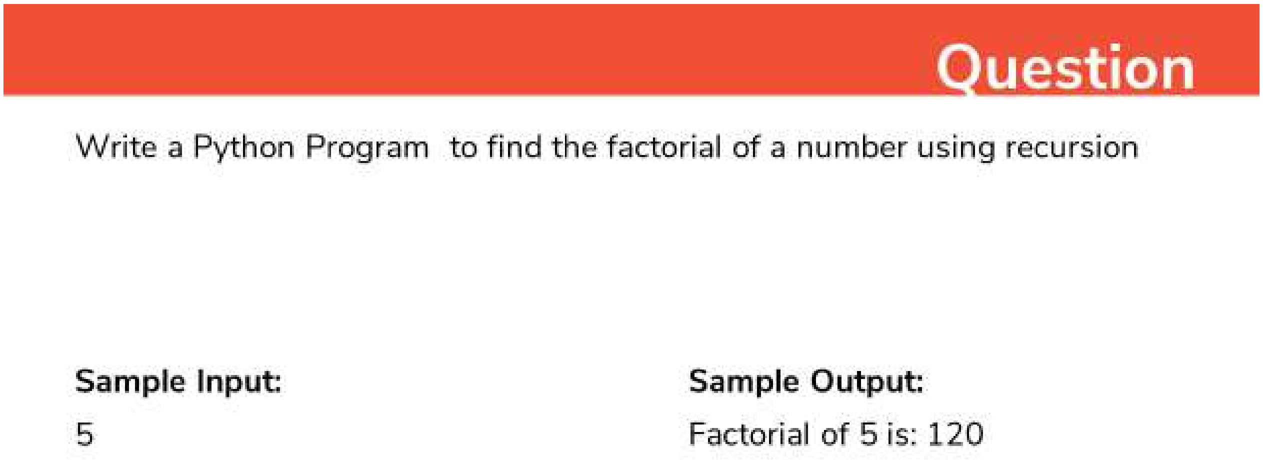
Recursion

Function calling itself is known as Recursion

Recursion is a backtracking process

Recursion internally uses stack to store all function calls

Base case is mandatory to terminate the recursion process.



Question

Write a Python Program to find the factorial of a number using recursion

Sample Input:

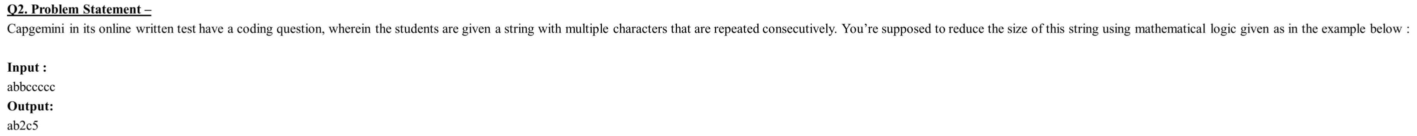
5

Sample Output:

Factorial of 5 is: 120

```
#Python program to find factorial of a number using recursion
def fact(n):
  #Base case
  if(n == 0 or n == 1):
    return 1
  else:
    return n * fact(n - 1)
#driver code
n = int(input("Enter the number: "))
print( fact(n) )
```

```
Enter the number: 6
720
```



Q2. Problem Statement –

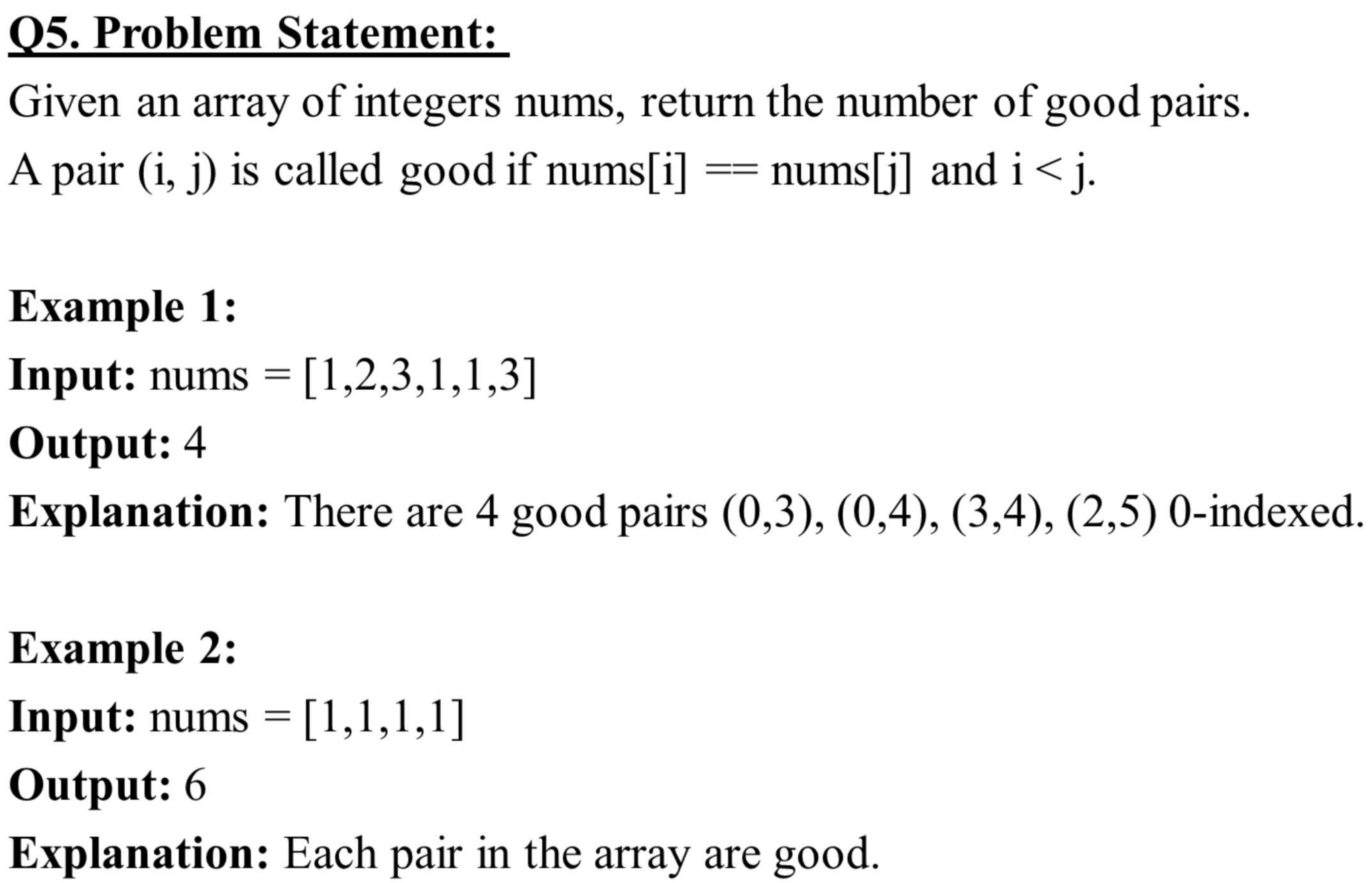
Capgemini in its online written test have a coding question, wherein the students are given a string with multiple characters that are repeated consecutively. You’re supposed to reduce the size of this string using mathematical logic given as in the example below :

Input :
abbccccc
Output:
ab2c5

```
#character count in a string
def char_count(words):
  unique = ""
  ans = ""
  for character in words:
    if character not in unique:
      #unique = unique + character
      unique += character
      count = words.count(character)
      if(count > 1):
        ans += character + str(count)
      else:
        ans += character
  return ans

#driver code
words = input()
print(char_count(words))
```

```
abbccccc
ab2c5
```



Q5. Problem Statement:

Given an array of integers nums, return the number of good pairs.

A pair (i, j) is called good if nums[i] == nums[j] and i < j.

Example 1:

Input: nums = [1,2,3,1,1,3]

Output: 4

Explanation: There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.

Example 2:

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

Output: 6

Explanation: Each pair in the array are good.

```
#Good pairs
def good_pairs(nums):
  count = 0
  for i in range(0, len(nums)):
    for j in range(i + 1, len(nums)):
      if(nums[i] == nums[j] and i < j):
        #count = count + 1
        count += 1
  return count
#driver code
nums = list(map(int, input().split(' ')))
print(good_pairs(nums))
```

```
1 1 1 1
6
```

Q6. Problem Statement:

Given an array of integers nums and an integer target, return **indices** of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.

Example 1:
Input: nums = [2,7,11,15], target = 9
Output: 0 1
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

```
#target_sum
def target_sum(nums, target):
  for i in range(0, len(nums)):
    for j in range(i + 1, len(nums)):
      if(nums[i] + nums[j] == target):
        return [i, j]

  return []
#driver code
nums = list(map(int, input().split(' ')))
target = int(input())
print(target_sum(nums, target))
```

Exceptional Handling

```
#Example 1

try:
    # code that might raise an exception
    result = 10 / 0
except Exception as e:
    # handle any exception
    print("An error occurred:", e)

```

```
An error occurred: division by zero
```

```
#Example 2

try:
    # code that might raise an exception
    result = 10 / 0
except ZeroDivisionError:
    # handle the specific exception (ZeroDivisionError in this
    print("Division by zero is not allowed.")
finally:
    # code that always runs, regardless of whether an exceptior
    print("This will always execute.")

```

```
Division by zero is not allowed.
This will always execute.
```

```
#Example 3

try:
    my_dict = {"a": 1, "b": 2}
    print(my_dict["c"])
except KeyError:
    print("Key not found in dictionary.")

```

```
Key not found in dictionary.
```

Modules

```python
#math module

import math
# Calculate the square root of a number
num = 25
square_root = math.sqrt(num)
print("Square root of {0} is: {1}".format(num, square_root))

# Calculate the factorial of a number
num = 5
factorial = math.factorial(num)
print("Factorial of {0} is: {1}".format(num, factorial))
```

```
Square root of 25 is: 5.0
Factorial of 5 is: 120
```

datetime module

```python
#date and time module

import datetime
# Get the current date and time
current_datetime = datetime.datetime.now()
print("Current date and time:", current_datetime)
# Get the current date
current_date = datetime.date.today()
print("Current date:", current_date)
# Get the current time
current_time = datetime.datetime.now().time()
print("Current time:", current_time)
```

```
Current date and time: 2024-02-17 09:48:31.000156
Current date: 2024-02-17
Current time: 09:48:31.002889
```

Prime number program

```
#Idea - 1 (for loop : 1 to n)
def is_prime1(n):
  count = 0
  for value in range(1, n + 1):
    if(n % value == 0):
      count += 1
  if(count == 2):
    print("prime")
  else:
    print("Not prime")

#driver code
n = int(input("Enter n value: "))
is_prime1(n)

```

```
Enter n value: 73
prime
```

```
#Idea - 2 (for loop : 2 to n - 1)
def is_prime2(n):
  count = 0
  for value in range(2, n):
    if(n % value == 0):
      count += 1

  if(count == 0 and n > 1):
    print("prime")
  else:
    print("Not prime")

#driver code
n = int(input("Enter n value: "))
is_prime2(n)

```

```
Enter n value: 5
prime
```

```
#Idea - 3 (for loop : 2 to n//2)
def is_prime3(n):
  count = 0
  for value in range(2, n//2 + 1):
    if(n % value == 0):
      count += 1

  if(count == 0 and n > 1):
    print("prime")
  else:
    print("Not prime")

#driver code
n = int(input("Enter n value: "))
is_prime3(n)

```

```
#Idea - 4 (for loop : 2 to sqrt(n))
import math
def is_prime4(n):
  count = 0
  for value in range(2, int(math.sqrt(n)) + 1 ):
    if(n % value == 0):
      count += 1

  if(count == 0 and n > 1):
    print("prime")
  else:
    print("Not prime")

#driver code
n = int(input("Enter n value: "))
is_prime4(n)

```

File handling

```
# Function to write some data to a file
```

```
def write_to_file(file_path):
    with open(file_path, "w") as file:
        file.write("Hello, world!\n")
        file.write("This is a sample file.\n")
        file.write("Python file handling is fun!\n")
```