# Writing Pythonic code

The following code is harmful

```
result_list = ['True', 'False', 'File not found']
result_string = ''
for result in result_list:
  result_string += result
result_string
```

```
'TrueFalseFile not found'
```

## sing join function as per PEP 8 rule



The following code is idiomatic

```
result_list = ['True', 'False', 'File not found']
result_string = ''.join(result_list)
result_string
```

```
'TrueFalseFile not found'
```

# Chain string functions

The following code is harmful

```
book_info = ' The Three Musketeers: Alexandre Dumas'
formatted_book_info = book_info.strip()
formatted_book_info = formatted_book_info.upper()
```

```
formatted_book_info = formatted_book_info.replace(':', ' by')
formatted_book_info
```

```
'THE THREE MUSKETEERS by ALEXANDRE DUMAS'
```

The following code is idiomatic

```
book_info = ' The Three Musketeers: Alexandre Dumas'
formatted_book_info = book_info.strip().upper().replace(':', ' by')
formatted_book_info
```

```
'THE THREE MUSKETEERS by ALEXANDRE DUMAS'
```

# Removing Duplicates from a List

Saved successfully!

```
ints_list = [1, 2, 3, 4, 3, 2]
temp = []
for x in ints_list:
    if x not in temp:
        temp.append(x)
ints_list = temp
print(f'Updated List after removing duplicates = {temp}')
```

```
Updated List after removing duplicates = [1, 2, 3, 4]
```

The following code is idiomatic using set()

```
ints_list = [1, 2, 3, 4, 3, 2]

ints_list1 = list(set(ints_list))
```

```
print(ints_list1)  # [1, 2, 3, 4]
```

```
[1, 2, 3, 4]
```

# some of the important functions in Python

## zip() in Python

The purpose of zip() is to map the similar index of multiple containers so that they can be used just using as single entity.

```
name = [ "Manjeet", "Nikhil", "Shambhavi", "Astha" ]
roll_no = [ 4, 1, 3, 2 ]
marks = [ 40, 50, 60, 70 ]

mapped = zip(name, roll_no, marks)

# converting values to print as set
mapped = set(mapped)

# printing resultant values
print ("The zipped result is : ",end="")
print (mapped)
```

```
The zipped result is : {('Manjeet', 4, 40), ('Shambhavi', 3, 60), ('Nikhil', 1, 50), ('Astha', 2, 70)}
```

Saved successfully!

How to unzip?

```
name,rollno,marks=zip(*mapped)
print(name,rollno,marks)
```

```
('Manjeet', 'Shambhavi', 'Nikhil', 'Astha') (4, 3, 1, 2) (40, 60, 50, 70)
```

# Map() function with lamda in Python

It is used t calls the specified function for each item of an iterable (such as string, list, tuple or dictionary) and returns a list of results.

```
def square(x):
    return x*x

numbers=[1, 2, 3, 4, 5]
sqrList=map(square, numbers)
print(list(sqrList))

# use lamda function to have direct values
                                    2, 3, 4])
```



```
[1, 4, 9, 16, 25]
[1, 4, 9, 16]
```

In the above example, the map() function applies to each element in the numbers[] list.

# Slicing operations in string

Program for exchanging first and last characters of a string using function

```
def change(string):
      return string[-1] + string[1:-1] + string[0]
string=input("Enter string:")
print("Modified string:",change(string))
print('original string :',string)
```

```
Enter string:vinay
Modified string: yinav
original string : vinay
```

## Removing one character from a string

```
def remove(string, n):
      first = string[:n]
      last = string[n+1:]
      return first + last
string=input("Enter the sring:")
n=int(input("Enter the index of the character to remove:"))
```

Saved successfully!

```
Enter the sring:ramajayam
Enter the index of the character to remove:4
Modified string:
ramaayam
```

## Sorting in Python using function

```
def myFunc(e):
  return len(e)
#myfunc=lambda x : len(x)
cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']
cars.sort(key=myFunc)
print(cars)
```

```
cars.sort(key=myFunc,reverse=True)
print(cars)

def myFunc(e):
  return e[1]

cars = [(1,'Ford'), (2,'Mitsubishi'), (3,'BMW'), (4,'VW')]
cars.sort(key=myFunc)
print(cars)

cars.sort(key=myFunc,reverse=True)
print(cars)
```

```
['VW', 'BMW', 'Ford', 'Mitsubishi']
['Mitsubishi', 'Ford', 'BMW', 'VW']
[(3, 'BMW'), (1, 'Ford'), (2, 'Mitsubishi'), (4, 'VW')]
), (1, 'Ford'), (3, 'BMW')]
```

Saved successfully!

# itertools groupby

```
from itertools import groupby

things = [("animal", "bear"), ("animal", "duck"), \
          ("plant", "cactus"), ("vehicle", "speed boat"),\
          ("vehicle", "school bus")]

for key, group in groupby(things, lambda x: x[0]):
    for thing in group:
      print ("A %s is a %s."%(thing[1], key))
```

```
A bear is a animal.
A duck is a animal.
A cactus is a plant.
```

## Combinations Of string "SADIK" OF SIZE 3.

```
from itertools import combinations

letters ="SADIK"

# size of combination is set to 3
a = combinations(letters, 3)
y = [' '.join(i) for i in a]
print(y)
```

Saved successfully!

```
object at 0x7fae5dc426a0>
'S D I', 'S D K', 'S I K', 'A D I', 'A D K', 'A I K', 'D I K']
```

## Permutations Of string "SADIK" OF SIZE 3.

```
from itertools import permutations

letters ="SADIK"

# size of permutaion is set to 3
a = permutations(letters, 3)
y = [' '.join(i) for i in a]
print(y)
```

['S A D' 'S A T' 'S A K' 'S D A' 'S D T' 'S D K' 'S T A' 'S T D' 'S T K' 'S K A' 'S K D' 'S K T' 'A S D'

## Accumulate()

```
# import the itertool module
# to work with it
import itertools

# import operator to work
# with operator
import operator

# creating a list GFG
GFG = [1, 2, 3, 4, 5]

# using the itertools.accumulate()
result = list(itertools.accumulate(GFG,
                                   operator.mul))

# printing each item from list
for each in result:
    print(each)
```

```
1
2
6
24
120
```

Saved successfully!

## Swaping by ^(Exclusive or)

```
a=5
b=4
a=a^b
```

```
b=a^b
a=a^b
print(a)
print(b)
```

```
4
5
```

Saved successfully!