1. Write a program to find the length of the string without using inbuilt function (len)

```python
def _len(iterable):
    _count = 0
    for item in iterable:
        _count += 1
    return _count

>>> _len('Hello')
5
>>> _len([1, 2, 3, 4])
4
>>>
>>> _len({1, 2, 3})
3
>>> _len((1, 2, 3, 4))
4
>>>
```

2. Write a program to reverse a string without using any inbuilt functions.

```python
def reverse(any_string):
    temp = []
    for i in range(len(any_string)-1, -1, -1):
        temp.append(any_string[i])
    return ''.join(temp)

>>> reverse('Hello world')
'dlrow olleH'
>>>
>>> reverse('Python')
'nohtyP'
>>>
>>> reverse('racecar')
'racecar'
>>>
```

3. Write a program to replace one string with another. e.g. "Hello World" replace "World" with "Universe".

```python
>>> s = 'Hello world'
```

```
>>> s.replace('world', 'universe')
'Hello universe'
>>>
```

4. How to convert a string to a list and vice-versa.

```python
def convert_to_string(any_list):
    return ''.join(any_list)

def convert_to_list(any_string):
    return any_string.split()

>>> convert_to_list('steve')
['steve']
>>> convert_to_string(['steve', 'jobs'])
'stevejobs'
```

5. Convert the string "Hello welcome to Python" to a comma separated string.

```python
>>> s = "Hello welcome to Python"
>>>
>>> s
'Hello welcome to Python'
>>>
>>> temp = s.split()
>>> temp
['Hello', 'welcome', 'to', 'Python']
>>> ','.join(temp)
'Hello,welcome,to,Python'
```

```python
>>> s = "Hello welcome to Python"
>>> words = s.split()
>>> print(*words, sep=',')
```

6. Write a program to print alternate characters in a string.

```python
>>> s = 'hello world'
>>> print(s[::2])  # Using Slicing Syntax
```

Hlowrd

7. Write a Program to print ascii values of the characters present in a string.

```python
>>> s = 'hello'
>>>
>>> for c in s:
        print(ord(c))
104
101
108
108
111
```

8. Write program to convert upper case to lower case and vice-versa without using inbuilt method.

```python
def convert(any_string):
    l = []
    for c in any_string:
        temp = ord(c)   # Get the ASCII value of the character
        if temp>= 97 and temp<=122:
            l.append(chr(temp - 32))
        elif temp>=65 and temp<=90:
            l.append(chr(temp+32))
    return ''.join(l)

>>> convert('Hello WorlD')
'hELLOwORLd'
```

```python
def convert(any_string):
    l = []
    for c in any_string:
        temp = ord(c)
        if temp in range(97, 123):
            l.append(chr(temp-32))
        elif temp in range(65, 91):
            l.append(chr(temp+32))
        else:
            l.append(chr(temp))
    return ''.join(l)
>>> convert('Hello WorlD')
'hELLOwORLd'
```

```
**9. Write program to swap two numbers without using 3rd variable.**
```python
>>> a = 10
>>> b = 20
>>>
>>> b, a = a, b
>>> a
20
>>> b
10
>>>
```
10. Write program to merge two different lists.
```python
>>> a = [1, 2, 3]
>>>
>>> b = [4, 5, 6]
>>>
>>> c = [*a, *b]
>>> c
[1, 2, 3, 4, 5, 6]
>>>
```
```python
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
```
```python
>>> a + b
[1, 2, 3, 4, 5, 6]
```
```python
# Using chain
>>> from itertools import chain
>>> s = chain(a, b)     # Returns an iterator
>>> list(s)
>>> [1, 2, 3, 4, 5, 6]
```
11. Write program to read a random line in a file. (ex. 50, 65, 78th line)
```python
from itertools import islice
def read_random_line(lineno):
```
```

```
    with open('Data/access-log.txt') as f:
        line = islice(f, lineno, lineno+1)
        return list(line)

print(read_random_line(2))
```
Alternate Solution
```python
def read_random_line(lineno):
    f = open('Data/sample.txt')
    for index, line in enumerate(f, start=1):
        if index == lineno:
            return line

print(read_random_line(10))
```

12. Write program to read a random lines in a file. (ex. I want read all lines 10th to 15th line)
```python
from itertools import islice
def read_n_lines(start_line, end_line):
    with open('Data/access-log.txt') as f:
        s = islice(f, start_line,end_line)
        for line in s:
            print(line)

read_n_lines(10, 15)
```
Alternate Solution
```python
def read_n_lines(start_line, end_line):
    with open('Data/access-log.txt') as enumerate(f, start=1):
        if index in range(start_line, end_line):
           print(line)

read_n_lines(10, 15)
```
13 Program to print last "N" lines of a file.
```python
from itertools import islice
def last_n_lines(n):
    with open('sample.txt') as f:
```

```
      for line in f:
         line_count += 1
      f.seek(0)
      lines = islice(f, line_count-n, None)
      return list(lines)
```

```python
from collections import
def tail(n):
    with open('sample.log') as f:
        d = deque(f, n)             # only last 'n' lines will be loaded to the
deque
        return d

last_10_lines = tail(10)     # returns last 10 lines of the file
for line in last_10_lines:
   print(line)
```

14. Write a program to check if the given string is Palindrome or not without using reversed method.

```python
>>> def is_palindrome(iterable):
       rev_iter = iterable[::-1]
       if iterable == rev_iter:
              return True
       else:
              return False

>>> is_palindrome('racecar')
True
>>> is_palindrome('malayalam')
True
>>> is_palindrome('hello')
False
>>>
```
15 Write a program to search for a character in a given string and return the corresponding index.

```python
```

```
>>> def search_character(string, key):
        for index, c, in enumerate(string):
                if c == key:
                        print(f'Character {c} is at index {index}')

>>> search_character('hello world', 'w')
Character w is at index 6
>>>
>>> search_character('hello world', 'd')
Character d is at index 10
>>>
>>>
```

```python
def search(string, key):
    if string.find(key) > -1:
        return string.find(key)
    else:
        print('Key not found')
```

```python
def search(string, key):
    try:
        return string.index(key)
    except ValueError:
        return "Key Not Found"
```

16 Write a program to get the below output

```python
sentence = "hello world welcome to python programming hi there"
d = {'h': ['hello', 'hi'], 'w': ['world', 'welcome'], 't': ['to', 'there'], 'p':
['python', 'programming'] }
from collections import defaultdict
d = defaultdict(list)
words = sentence.split()
for word in words:
   d[word[0]].append(word)
```

17 Write a to replace all the characters with - if the character occurs more than once in a string

```python
my_string = 'hellohai' # O/P should be '-e--o-ai'
```

```
my_string = 'hellohai'

new_string = ''.join(['-' if s.count(c) > 1 else c for c in my_string])
print(new_string)
```

18 write a decorator that returns only positive values of subtraction

```python
def positive(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return abs(result)
    return wrapper

@positive
def sub(a, b):
    return a - b

>>> sub(1, 2)
1
>>> sub(-1, 100)
101
>>> sub(-100, 1)
101
>>> sub(3, 1)
2
>>> sub(-1, -2)
1
>>> sub(-2, 1)
3
>>>
```

19 How to get the count of number of instances of a class that is being created.

```python
class Login:
   login_count = 0    # Class Variable that keeps count of login counts
   def __init__(self):
       Login.login_count += 1
```

```
>>> u1 = Login()
>>> Login.login_count
1
>>> u2 = Login()
>>> Login.login_count
2
>>> u3 = Login()
>>> Login.login_count
3
>>>
```

20 Write a function which takes a list of strings and integers.If the item is a string it should print as is and if the item is integer of float it should reverse it.

```python
def spam(items):
    for item in items:
        if isinstance(item, str):   # Check if the item is an instance of String
            print(item)
        else:
            temp = str(item)    # Typecast Integer to String
            print(temp[::-1])   # Reverse the String

>>> spam(['apple', 'yahoo', '1234', 100, 123.76, '26.23'])
apple
yahoo
1234
001
67.321
26.23
>>>
```

21 Write a class named Simple and it should have iteration capability.

```python
class Simple:
    def __init__(self, items):
        self._items = items

    def __iter__(self):
```

```
        return iter(self._items)

>>> s = Simple([1, 2, 3, 4, 5])
>>>
>>> for item in s:
       print(item)
1
2
3
4
5
>>>
```
22 Write a Custom class which can access the values of dictionaries using d['a'] and d.a

```python
class MyDict:
    def __init__(self, d):
        self._dict = d

    def __getitem__(self, item):
        return self._dict[item]

    # __getattr__ method is called only for missing attributes.(i.e. if you try to access an attribute which is not in instance dict)
    def __getattr__(self, name):
        return self._dict[name]

>>> d = MyDict({'a': 1, 'b': 2})
>>> d.a
1
>>> d.b
2
>>> d['a']
1
>>> d['b']
2
>>>
```

23 Write a python program to get the below output

```python
sentence = "Hi How are you"
o/p should be "iH woH era uoy"

>>> sentence = "Hi How are you"
>>> words = sentence.split()
>>> words
['Hi', 'How', 'are', 'you']
>>> reversed_words = [word[::-1] for word in words]
>>> reversed_words
['iH', 'woH', 'era', 'uoy']
>>> ' '.join(reversed_words)
'iH woH era uoy'
>>>
```

24 Write a python program to get the below output

```python
sentence = "Hi How are you"
o/p should be "ouy era woH iH"

>>> sentence = "Hi How are you"
>>> sentence[::-1]
'uoy era woH iH'
>>>
```

25 Write a lambda function to add two numbers (a, b)
```python
>>> add = lambda a, b: a + b
>>> add(1, 2)
3
>>> add(100, 300)
400
>>>
```
26 What is the output of the following
```python
a = [1, 2, 3]
b = [4, 5, 6]
print([a, b])
print((a, b))
```

```
>>> print((a, b))
([1, 2, 3], [4, 5, 6])   # Tuple of Lists
>>> print([a, b])
[[1, 2, 3], [4, 5, 6]]   # List of Lists
>>>
```

27 How to remove duplicates from the list without using inbuilt functions

```python
>>> items = [1, 2, 3, 4, 1, 2, 3, 4, 5]
>>> uniques = []
>>> for item in items:
        if item not in uniques:
                uniques.append(item)
>>> print(uniques)
[1, 2, 3, 4, 5]
>>>
```

28 Find the longest word in the sentence

```python
sentence = "Hello world. Welcome to Python"
>>> sentence = "Hello world. Welcome to Python"
>>> words = sentence.split()
>>> d = {word: len(word) for word in words}
>>> max(d.items(), key= lambda item: item[-1])
('Welcome', 7)
>>>
```

```python
sentence = "hello world welcome to programming"
words = sentence.split()
max_len = 0
max_word = ''

for word in words:
    if len(word) > max_len:
        max_len = len(word)
        max_word = word
print(max_len, max_word)
```

```python
```

```
>>> sentence = "hello world welcome to programming"
>>> max(sentence.split(), key=len)
>>> 'programming'
```

29 write a program to reverse the values in the dictionary if the value is of type String

```python
>>> d = {'a': 'hello', 'b': 100, 'c': 10.1, 'd': 'world'}
>>> rev = {  key: value[::-1] if isinstance(value, str) else value for key, value in d.items()}
>>> rev
{'a': 'olleh', 'b': 100, 'c': 10.1, 'd': 'dlrow'}
```

30 write a program to get 1234

```python
t = ('1', '2', '3', '4')
>>> t = ('1', '2', '3', '4')
>>> ''.join(t)   # Use join function
'1234'
>>>
```

31 How to get the elements that are in list b but not in list a

```python
a = [1, 2, 3] b = [1, 2, 3, 4]
>>> a = [1, 2, 3]
>>> b = [1, 2, 3, 4]
>>> set_a = set(a)      # Convert the list to set
>>> set_b = set(b)
>>> set_b.difference(set_a)
{4}
>>>
```

```python
a = [1, 2, 3]
b = [1, 2, 3, 4]

for item in b:
    if item not in a:
        print(item)
```

32 A function takes a variable number of positional arguments as input. How to check if the arguments that are passed are more than 5

```python
>>> def spam(*args):
        if len(args) > 5:
               print('Length of arguments passed is greater than 5')
        else:
               print('Length Argument passed is less than 5')


>>> spam(1, 2, 3, 4, 5, 6, 7)
Length of arguments passed is greater than 5
>>>
>>> spam(1, 2)
Length Argument passed is less than 5
>>>
>>> spam()
Length Argument passed is less than 5
>>>
>>> spam(1, 2, 3, 4, 5)
Length Argument passed is less than 5
>>>
```

33 Count the number of occurrences of "CRITICAL", "INFO" and "ERROR" lines in a log file.

```python
# Assume Below is the contents of the log file

lines = """CRITICAL:Hello world
INFO: This is an info
ERROR: This is an error
CRITICAL: This is critical
CRITICAL:Hello world
INFO: This is an info
ERROR: This is an error
CRITICAL: This is critical
CRITICAL:Hello world
INFO: This is an info
ERROR: This is an error
CRITICAL: This is critical
CRITICAL:Hello world
INFO: This is an info
ERROR: This is an error
```

```
CRITICAL: This is critical"""

>>> from collections import defaultdict
>>> _errors = defaultdict(int)
>>> for line in lines.split('\n'):    # Split the line based on newline character
        # Split each line based on ":" to separate out error message part.
       error_type, other = line.strip().split(':')
       _errors[error_type] +=1

>>> _errors
defaultdict(<class 'int'>, {'CRITICAL': 8, 'INFO': 4, 'ERROR': 4})
>>> _errors['INFO']
4
>>> _errors['ERROR']
4
>>> _errors['CRITICAL']
8
>>>
```

34 Write a function to reverse any iterable without using reverse function.

```python
>>> a = [1, 2, 3, 4, 5]
>>> _reversed = []
>>> for i in range(len(a)-1, -1,-1):
       _reversed.append(a[I])

>>> _reversed
[5, 4, 3, 2, 1]
>>>
```

35 Write a function to print the output below.

```python
# func("TRACXN", 0)  # Should print RCN
# func("TRACXN", 1)  # Should print TAX

>>> def func(string, flag):
       if flag:
              return string[0::2]
       return string[1::2]

>>> func('TRACXN', 0)
```

```
'RCN'
>>> func('TRACXN', 1)
'TAX'
>>>
```

```python
def func(string, flag):
    return string[::2] if flag else string[1::2]
```

36 Sum all the numbers in the below string.

```python
import re
s = "Sony12India567Pvt2ltd"
total = 0.00
>>> r = re.findall(r'[\d]', s)
>>> r
['1', '2', '5', '6', '7', '2']
>>> for item in r:
       total += int(item)
>>> total
23.0
```

37 Write a program to sum all the numbers in below string.

```python
import re
s = "Sony12India567Pvt2ltd" # eg.12+567+2

>>> rr = re.findall(r'[\d]+', s)
>>> rr
['12', '567', '2']
>>> total = 0.00
>>> for item in rr:
       total += int(item)
>>> total
581.0
```

38 Print all the numbers in the below list

```python
a = ['abc', '123', 'hello', '23']
```

```
>>> for item in a:
      if item.isnumeric():
        print(item)

123
23
```

39 Program to print the number of occurrences of characters in a String without using inbuilt functions.

```python
>>> s = 'helloworld'
>>> from collections import defaultdict
>>> d = defaultdict(int)
>>> for c in s:
       d[c] +=1
>>> print(d)
defaultdict(<class 'int'>, {'h': 1, 'e': 1, 'l': 3, 'o': 2, 'w': 1, 'r': 1, 'd': 1})
```

40 Program to print only the repeated characters and count of the same.

```python
>>> s = 'helloworld'
>>> from collections import defaultdict
>>> d = defaultdict(int)
>>> for c in s:
       d[c] +=1
>>> for key, value in d.items():
      if value > 1:
        print(key, value)
```

```python
d = {ch: s.count(ch) for ch in s if s.count(ch) > 1}
```

41 Write a program to get alternate characters of a string in list format.

```python
>>> s = 'hello world welcome to python'
>>> alternate_chrs = [ c for c in s[::2]]
>>> alternate_chrs
['h', 'l', 'o', 'w', 'r', 'd', 'w', 'l', 'o', 'e', 't', ' ', 'y', 'h', 'n']
```

42 Write a program to get square of list of number's using lambda function .

```python
>>> a = [1, 2, 3, 4, 5]
>>> squares = lambda number: number ** 2
>>> b = [ squares(item) for item in a]
>>> b
[1, 4, 9, 16, 25]
```

43 Write a function that accepts two strings and returns True if the two strings are anagrams of each other.

```python
def is_anagram(str1, str2):
    if str1.upper() == str2.upper():    # Return False if both words are same
        return False
    s_str1 = sorted(str1.upper())   # Convert to upper case and sort
    s_str2 = sorted(str2.upper())
    if s_str1 == s_str2:    # Return True if both the lists are same
        return True
    else:
        return False
>>> is_anagram('ate', 'eat')
True
>>> is_anagram('racecar', 'racecar')
False
>>> is_anagram('file', 'life')
True
>>> is_anagram('hello', 'world')
False
>>> is_anagram("sinks", "skin")
False
>>> is_anagram("Listen", "silent")
True
```

44 Write a program to iterate through list and build a new list, only if the items of the list has even number of characters.

```python
>>> names = ['apple', 'yahoo', 'google', 'gmail', 'walmart', 'flipkart',
'facebook', 'amazon']
>>> [ name for name in names if len(name) % 2 == 0]
```

```
['google', 'flipkart', 'facebook', 'amazon']
```

45 Write a program to iterate through list and build a new dictionary, only if the items of the list has even number of characters.

```python
>>> names
['apple', 'yahoo', 'google', 'gmail', 'walmart', 'flipkart', 'facebook', 'amazon']
>>> { name: len(name) for name in names if len(name) % 2 == 0}
{'google': 6, 'flipkart': 8, 'facebook': 8, 'amazon': 6}
```

46 Write a program which squares the numbers in a list using map object

```python
# Squares of numbers using map
# map is used to map a function with a iterable
a = [1, 2, 3, 4, 5]

def squares(item):
    return item ** 2

# Returns a map object, which happens to be an iterator.
m = map(squares, a)

# To get the squares of numbers, feed the map object to for loop
for item in m:
    print(item)

# Mapping lambda function to map object
m = map(lambda item: item ** 2, a)
```

47 Count number of lines in a file without loading the file to the memory

```python
# Counting number of lines in a file without loading the file to the memory
with open('sample.txt') as f:
    _count = 0
    # Iterate over a file object and increment the _count
    for line in f:
        _count +=1

print(f'No of Lines: {_count}')
```

48 Printing line and line no's

```python
with open('sample.txt') as f:
    for line_no, line in enumerate(f, start=1):
        print(line_no, line)
```

49 Write a Program to print the sum of entire list and sum of only internal list

```python
l = [[1,2,3],[4,5,6],[7,8,9]]
# Add the contents of internal list. ([6, 15, 24])
sum_internal = [sum(item) for item in l]

# Add the contents of entire list. (45)
sum_iternal = [sum(item) for item in l]
sum_whole_list = sum(sum_internal)
```

```python
items = [[1,2,3],[4,5,6],[7,8,9]]
# Using List Comprehension
total = sum([each_item for each_internal_list in items for each_item in each_internal_list])
```

50 Write a program to reverse the list as below

```python
words = ["hi", "hello", "python"]
# o/p ['nohtyp', 'olleh', 'ih']
words = words[::-1]
words = [word[::-1] for word in words]
```

51 Write a program to update the tuple

```python
t1 = (1, 2, 3, 4)
t2 = (100, 200, 300)
# o/p (1, 2, 3, 4, 100, 200, 300)
t = t1 + t2
```

52 Write a program to replace value present in nested dictionary.

```python
```

```
d = {'a': 100, 'b': {'m': 'man', 'n': 'nose', 'o': 'ox', 'c': 'cat'}}
# Replace "nose" with "net"
for key, value in d.items():
    if isinstance(value, dict):
        d[key]['n'] = "net"
print(d)
```

53 Write a program to count the number of white spaces in a file.

```python
import re
white_spaces = 0
with open('data/sample.txt') as f:
    for line in f:
        match = re.findall(r'\s', line)
        if match:
            white_spaces += len(match)
print(white_spaces)
```

54 Grouping anagrams.

```python
>>> from collections import defaultdict
>>> words = ['eat', 'ate', 'tea', 'hello', 'silent', 'listen']
>>> d = defaultdict(list)
>>> for word in words:
       s = ''.join(sorted(word))
       d[s].append(word)

>>> print(d)
defaultdict(<class 'list'>, {'aet': ['eat', 'ate', 'tea'], 'ehllo': ['hello'],
'eilnst': ['silent', 'listen']})
>>> group = list(d.values())
>>> group
[['eat', 'ate', 'tea'], ['hello'], ['silent', 'listen']]
>>>
```

55 What is the difference between defaultdict and normal dictionary.

```python
"""
```

Defaultdict
---
1. When each key is encountered for the first time, it will not be there in the mapping.
2. So an entry is automatically created with default value (an empty list in case of defaultdict of list and zero in case of defaultdict int).
3. When keys are encountered again, the look-up proceeds normally as like a normal dictionary.
4. So, in defaultdict, creation of key, initialisation will happen simultaneously.

Normal Dictionary
---
1. In case of normal dictionary, if the key does not exist, "KeyError" is raised.
2. In order to work on the value, first the key needs to be created and initialised.
"""
```

**56 Explain property decorator in python.**
```python
#
```

57 What is Mutable and Immutable datatypes.
```python
"""
1. Mutable datatypes are objects whose value can be changed after creation. e.g. list, dict, set, user defined classes.
2. Immutable datatypes are objects whose value can not be changed after creating. e.g. int, float, bool, tuple, namedtuple
"""
```

58 Explain get() method in dictionaries.
```python
"""
point =  {'a': 1, 'b': 2}
1. Values of dictionary can be accessed in two different ways. using square bracket syntax and the other one is using get() method.
2. When we try to access a key of a dictionary which does not exist using square bracket syntax (point['c']), "KeyError" exception is raised.
3. When we try to access a key of a dictionary which does not exist using get() method (point.get('c')), None is returned and no exception is raised.
```
```

4. We can pass a positional argument to get() method as custom message, so that get() method returns the custom message if the key does not exist.
           e.g. profile.get('c', 'Sorry the key does not exist')
"""
```

**59 Write a list comprehension to get a list of even numbers from 1-50**
```python
evens = [ item for item in range(1, 51) if item % 2 == 0]
```

60 Find the longest non-repeated substring in the below string
```python
>>> s = "This is a Programming language and Programming is fun"
# make dictionary with word and its length pair for only those words which are not repeated.
>>> d = { word: len(word) for word in s.split() if s.count(word) == 1}
>>> d
{'This': 4, 'language': 8, 'and': 3, 'fun': 3}
# Sort the dictionary based on values and get the last item
>>> sorted(dd.items(), key=lambda item: item[-1])[-1]
('language', 8)
```

61 Write a program to find the duplicate elements in the list without using inbuilt functions
```python
names = ['apple', 'google', 'apple', 'yahoo', 'google']

unique_items = set(names)

for item in unique_items:
    _count = 0
    for name in names:
        if item == name:
            _count += 1
    if _count > 1:
        print(item)
```

62 Write a program to count the number occurrences of each item in the list without using any inbuilt functions
```python

```
names = ['apple', 'google', 'apple', 'yahoo', 'google', 'facebook', 'gmail',
'yahoo']

# Get the unique elements present in the list
unique_items = set(names)

# declare an empty dictionary
d = {}

for item in unique_items:
    _count = 0
    for name in names:
        if item == name:
            _count += 1
    d[item] = _count
```
**63 Write a function to check if the number is Prime**
```python
>>> def is_prime(number):
# any is a builtin function that returns True if any one item in iterable evaluates
to  boolean True
       return not any([number % 2 == 0 for i in range(2, number)])

>>> is_prime(10)
False
>>> is_prime(11)
True
>>> is_prime(13)
True
>>> is_prime(1)
True
>>> is_prime(2)
True
>>> is_prime(3)
True
>>> is_prime(4)
False
>>> is_prime(5)
True
>>> is_prime(6)
False
>>>
```

64 How to create a tuple using range function

```python
>>> tuple(range(10))
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

65 Write a program to find the largest number in the list without using any inbuilt functions

```python
>>> numbers = [10, 20, 30, 40, 50]
>>> largest = 0
>>> for item in numbers:
        if item > largest:
                largest = item
>>> print(largest)
50
```

66 Write a method that returns the last digit of an integer. For example, the call of get_last_digit(3572) should return 2.

```python
>>> def get_last_digit(number):
        temp = str(number)    # Typecast Integer to String
        return int(temp[-1])  # Return the last digit

>>> get_last_digit(1234)
4
>>> get_last_digit(934)
4
>>> get_last_digit(1)
1
>>>
```

67 Write a program to find most common words in a given list.

```python
words = [
'look', 'into', 'my', 'eyes', 'look', 'into', 'my', 'eyes',
'the', 'eyes', 'the', 'eyes', 'the', 'eyes', 'not', 'around', 'the', 'eyes',
"don't", 'look', 'around', 'the',
'eyes', 'look', 'into','my', 'eyes', "you're", 'under'
]
```

```
>>> from collections import Counter
>>> c = Counter(words)
>>> c
Counter({'eyes': 8, 'the': 5, 'look': 4, 'into': 3, 'my': 3, 'around': 2, 'not': 1,
"don't": 1, "you're": 1, 'under': 1})
>>> c.most_common()
[('eyes', 8), ('the', 5), ('look', 4), ('into', 3), ('my', 3), ('around', 2),
('not', 1), ("don't", 1), ("you're", 1), ('under', 1)]
>>>
```

```python
>>> d = {word: words.count(word) for word in words}
>>> d
{'look': 4, 'into': 3, 'my': 3, 'eyes': 8, 'the': 5, 'not': 1, 'around': 2,
"don't": 1, "you're": 1, 'under': 1}

>>> sorted(d.items(), key= lambda item: item[-1])    # Sort the Dictionary by its
value
[('not', 1), ("don't", 1), ("you're", 1), ('under', 1), ('around', 2), ('into', 3),
('my', 3), ('look', 4), ('the', 5), ('eyes', 8)]
>>>
```

```python
>>> from collections import defaultdict
>>> d = defaultdict(int)
>>> for word in words:
        d[word]+=1
>>> d
defaultdict(<class 'int'>, {'look': 4, 'into': 3, 'my': 3, 'eyes': 8, 'the': 5,
'not': 1, 'around': 2, "don't": 1, "you're": 1, 'under': 1})
>>> sorted(d.items(), key = lambda item: item[-1])
[('not', 1), ("don't", 1), ("you're", 1), ('under', 1), ('around', 2), ('into', 3),
('my', 3), ('look', 4), ('the', 5), ('eyes', 8)]
>>>
```

68 Make a function named tail that takes a sequence (like a list, string, or tuple) and a number n and returns the last n elements from the given sequence, as a list.

```python
def tail(iterable, n):
    if not isinstance(n, int):
        raise TypeError('Value of N should be Positive Integer')
    if n <=0:
        return []
    return list(iterable)[-n:]

>>> tail([1, 2, 3, 4, 5], 3)
[3, 4, 5]
>>> tail('hello', 2)
['l', 'o']
>>> tail('hello', 0)
[]
```

69 Write function named is_perfect_square that accepts a number and returns True if it's a perfect square and False if it's not.

```python
>>> from math import sqrt
>>> def is_perfect_square(number):
      return number == sqrt(number) ** 2

>>> is_perfect_square(5)
False
>>> is_perfect_square(4)
True
>>> is_perfect_square(9)
True
>>> is_perfect_square(5)
False
>>>
```

70 Write a program to get all the duplicate items and the number of times the item is repeated in the list.

```python
>>> names = ['apple', 'google', 'apple', 'yahoo', 'yahoo', 'facebook', 'apple', 'gmail', 'gmail', 'gmail', 'gmail']
>>> unique_names = set(names)
>>>
>>> { name: names.count(name) for name in unique_names if names.count(name) > 1}
{'apple': 3, 'gmail': 4, 'yahoo': 2}
```

```
>>>
```

71 Write a program to count the number of occurrences of each word in a file.

```python
from collections import defaultdict

with open("Data/sample.txt") as f:
    # declare a default dictionary
    d = defaultdict(int)
    # Iterate over each line in the file object
    for line in f:
        # Check if the line has atleast one character
        if line.strip():
            # Split the line on white space
            words = line.split()
            # Iterate through each word in words list and build a dictionary with
word and count pair
            for word in words:
                d[word] += 1
```

72 Write a program to count the number of occurrences of vowels in a file.

```python
from collections import defaultdict

with open('Data/code.txt') as f:
    d = defaultdict(int)
    # Iterate over each line in the file object
    for line in f:
        # Check if the line has atleast one character
        if line.strip():
            # Get each character
            for c in line:
                # Check if that character is vowel or not
                if c in 'aeiou':
                    d[c] += 1
```

73 Write a program to print all numeric values in a list

```python

items = ['apple', 1.2, 'google', '12.6', 26, '100']
```

```
for item in items:
    if isinstance(item, (int, float)):
        print(item)
```

74 Triangle Patterns

```python
# Left Justified Triangle
for i in range(1, 6):
    print(f"{'*'*i:<5}")
```

```
*
* *
* * *
* * * *
* * * * *
```

```
# Right Justified Triangle
for i in range(1, 6):
    print(f"{'*'*i:>5}")
```

```
        *
      * *
    * * *
  * * * *
* * * * *
```

```
# Equilateral Triangle
for i in range(1, 6):
    print(f"{'* '*i:^10}")
```

```
    *
   * *
  * * *
 * * * *
* * * * *
```

```
# Inverted Triangles (Left Justified)
for i in range(6, 0, -1):
    print(f"{'*'*i:<6}")
```

```
* * * * * *
* * * * *
```

```
* * * *
* * *
* *
*

# Inverted Triangles (Right Justified)
for i in range(6, 0, -1):
    print(f"{'*'*i:>12}")

* * * * * *
  * * * * *
    * * * *
      * * *
        * *
          *
# Inverted Triangles (Centre)
for i in range(6, 0, -1):
    print(f"{'* '*i:^12}")

 * * * * * *
  * * * * *
   * * * *
    * * *
     * *
      *

# Number Pattern in triangle (Left Justified)

pat = ''
for i in range(1, 6):
    pat += str(i)
    print(f'{pat:<5}')

1
12
123
1234
12345

# Number Pattern in triangle (Right Justified)
pat = ''
for i in range(1, 6):
    pat += str(i)
```

```
    print(f'{pat:>5}')

    1
   12
  123
 1234
12345

# Number Pattern in triangle (Centre)
pat = ''
for i in range(1, 6):
    pat = pat + ' '+ str(i)
    print(f'{pat:^10}')

     1
    1 2
   1 2 3
  1 2 3 4
 1 2 3 4 5
```

75 Write a program count the occurrence of a particular word in the file

```python
def count_words(word):
    _count = 0
    with open('Data/code.txt') as f:
        for line in f:
            _count += line.count(word)
    return _count
```

```python
import re

def count_words(word):
    _count = 0
    with open('Data/code.txt') as f:
        for line in f:
            _count +=  len(re.findall(word, line))
    return _count
```

76 Write a program to map a product to a company and build a dictionary with company and list of products pair

```python
>>> all_products = ['iPhone', 'Mac', 'Gmail', 'Maps', 'iWatch', 'Windows',
                'iOS', 'Google Drive', 'One Drive']

>>> # Pre-defined products for different companies
>>> apple_products = {'iPhone', 'Mac', 'iWatch'}
>>> google_products = {'Gmail', 'Maps', 'Google Drive'}
>>> msft_products = {'Windows', 'One Drive'}

>>> from collections import defaultdict

>>> d = defaultdict(list)

>>> for product in all_products:
      if product in apple_products:
         d['Apple Inc'].append(product)
      elif product in google_products:
         d['Google'].append(product)
      elif product in msft_products:
         d['Microsoft'].append(product)

>>> defaultdict(<class 'list'>, {'Apple Inc': ['iPhone', 'Mac', 'iWatch'],
'Google': ['Gmail', 'Maps', 'Google Drive'], 'Microsoft': ['Windows', 'One
Drive']})
```

77 Write a program to rotate items of the list

```python
>>> names = ["apple", "google", "yahoo", "gmail", "facebook", "flipkart", "amazon"]

>>> def rotate(iterable, n):
       for _ in range(n):
          f = iterable.pop()
          iterable.insert(0, f)
       return iterable

>>> rotate(names, 1)
['amazon', 'apple', 'google', 'yahoo', 'gmail', 'facebook', 'flipkart']
>>> rotate(names, 2)
['facebook', 'flipkart', 'amazon', 'apple', 'google', 'yahoo', 'gmail']
>>> rotate(names, 3)
```

```
['google', 'yahoo', 'gmail', 'facebook', 'flipkart', 'amazon', 'apple']
```

```python
>>> from collections import deque
>>> def _rotate(iterable, n):
      d = deque(iterable)
      d.rotate(n)
      return list(d)

>>> _rotate(names, 1)
>>> ['amazon', 'apple', 'google', 'yahoo', 'gmail', 'facebook', 'flipkart']
>>>
>>> _rotate(names, 2)
>>> ['flipkart', 'amazon', 'apple', 'google', 'yahoo', 'gmail', 'facebook']
```

78 Write a program to rotate characters in a string
```python
>>> def rotate_string(string, n):
      string = list(string)
      for _ in range(n):
        f = string.pop()
        string.insert(0, f)
      return ''.join(string)

>>> rotate_string("helloworld", 1)
>>> dhelloworl
>>> rotate_string("helloworld", 1)
>>> ldhellowor
```

```python
# Rotating words in a sentence
>>> sentence = "Hello world welcome to python"
>>> words = sentence.split()

>>> def rotate(iterable, n):
      for _ in range(n):
        f = iterable.pop()
        iterable.insert(0, f)
      return ' '.join(iterable)
>>> rotate(words, 1)
```

```
>>> python Hello world welcome to
>>> rotate(words, 2)
>>> to python Hello world welcome
```

79 Write a program to count the number of white spaces in a given string
```python
>>> import re

>>> sentence = """Hello world welcome to Python Hi  How are you. Hi how are you"""
>>> spaces = re.findall(r'\s', sentence)

>>> print(len(spaces))
>>> 13
```
80 Write a program to print only non-repeated characters in a string
```python
>>> s = 'helloworld'
>>> r = [ c for c in s if s.count(c) == 1]
>>> print(r)
>>> ['h', 'd', 'w', 'e', 'r']
```

81 What is the difference between a list and a tuple
```python
"""
1. The main difference between a list and a tuple is that list's are mutable and
tuples are immutable.
2. Python over allocates memory for lists. The reason for over allocation of memory
is to support append operation.
Where as in tuples, memory is not over allocated, as tuples does not support append
operation.
(Since tuples are immutable, it does not support append operation).
3. Tuples are more memory efficient than lists. (because in tuples no extra memory
is allocated. It is fixed).
4. Tuples are negligibly faster than lists.
"""
```

82 Write a program to print all the consonants in a given string
```python
>>> s = 'helloworld'
```

```
>>> consonants = [c for c in s if not c in 'aeiou']
>>> print(consonants)
>>> ['h', 'l', 'l', 'w', 'r', 'l', 'd']
```

83 Write a program to count the number of commented lines in a text file
```python
>>> with open('Data/code.txt') as f:
      _count = 0
      for line in f:
        if line.strip():
           if line.startswith('#'):
              _count += 1
>>> print(f"No of Commented lines {_count}")
```

84 Write a program to check if the year is leap year or not
```python
>>> import calendar
>>> print(calendar.isleap(2012))
>>> True
>>> print(calendar.isleap(2013))
>>> False
>>> print(calendar.isleap(2016))
>>> True
**85 Liner Search**
```python
>>> a = list(range(20))
>>> def _search(iterable, key):
      return any(item == key for item in iterable)
>>> print(_search(a, 17))
True
>>> print(_search(a, 21))
False
```

```python
# Alternate Solution
def _search(iterable, key):
    found = False
    for item in iterable:
        if item == key:
            found = True
```
```

```
            break
    if found:
        return True
    else:
        return False
```

86 Difference between xrange and range
```python
"""
python2- xrange
python3- range

1. xrange does not stop, start and step attributes. But range object has start,
stop and step attributes.
  Python-3
  r = range(0, 10)
  r.start
  r.stop
  r.step

  r = xrange(0, 10)
  In Python-2 The above attributes are not supported.

2. range Object supports slicing! But xrange does not support slicing

3. range object has __contains__ method implemented. So it supports membership
testing.
   But xrange does not implement __contains__ method.
   So Python will iterate over each and every item in the range xrange object until
it finds a match.
   (So if you are searching for a number in range is faster than xrange)

4. range will accept integer of any size. But xrange objects accepts integer size
equivalent to C long!
"""
```
87 Write a program to count no of capital letters in a string
```python
>>> from collections import defaultdict

>>> sentence = "Hi How are You WelCome to PytHon"
>>> capital_letters = defaultdict(int)
```

```
>>> for c in sentence:
      if ord(c) >= 65 and ord(c) <= 90:
        capital_letters[c] += 1
>>> print(capital_letters)
>>> defaultdict(<class 'int'>, {'H': 3, 'Y': 1, 'W': 1, 'C': 1, 'P': 1})
```

88 Write a program to get the below output

```python

>>> for i in range(1, 5):
      print('* '*1)
      j = i + 1
      print('* '*j)

*
* *
*
* * *
*
* * * *
*
* * * * *
```

89 Write a program to get the below output

```python
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> for i in range(0, len(a), 2):
      print(a[i:i+2])
>>> [1, 2]
    [3, 4]
    [5, 6]
    [7, 8]
    [9]
```

90 Write a program to check if the elements in the second list is series of continuation of the items in the first list

```python
```

```
a = [10, 12, 14, 16, 18]
b = [20, 22, 24, 26, 28]

a = [0, 5, 10, 15]
b = [20, 25, 30, 35, 40]

x = [10, 20, 30, 40]
y = [50, 40, 60, 70]

>>> def _series(iter1, iter2):
      # Get the difference of the series
      diff = a[1]-a[0]
      # Combine both the lists
      c = iter1 + iter2
      return all([True if c[i] + diff ==  c[i+1] else False for i in range(0,
len(c)-1)]

>>> print(_series(a, b))
>>> True
>>> print(_series(x, y))
>>> False
```

91 What is the difference between append() and extend() method in list

```python
"""
1. append() method appends one item at the end of the list.
2. extend() method appends all the items of the iterable to the end of the list.
3. Both append() and extend() method's mutates the existing list.

e.g.
>>> a = [1, 2, 3]
>>> b = (4, 5, 6)
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6]

>>> c = {7, 8, 9}
>>> a.extend(c)
>>> a
[1, 2, 3, 4, 5, 6, 8, 9, 7]
```

```
>>> d = {'a': 1, 'b': 2, 'c': 3}
>>> a.extend(d)
>>> a
[1, 2, 3, 4, 5, 6, 8, 9, 7, 'a', 'b', 'c']

The list "a" is getting mutated each time when it is extended.
"""
```

92 Write a program to find the first repeating character in a string

```python
>>> s = 'helloworld'
>>> for c in s:
      if s.count(c) > 1:
        print(c)
        break
>>> l
>>> s = 'heloworldd'
>>> l
>>> s = 'heoworldd'
>>> o
```

93 Write a program to find the index of nth occurrence of a sub-string in a string

```python

>>> sentence = "hello world welcome to python hello hi how are you hello there"

>>> import re

>>> def index_nth_occurance(sentence, pat, n):
      matches = re.finditer(pat, sentence)
      _count = 0
      for match in matches:
         _count +=1
         if _count == n:
            return f"Start Index: {match.start()}, End Index: {match.end()}"

>>> index_nth_occurance(sentence, 'hello', 3)
>>> Start Index: 51, End Index: 56
>>> index_nth_occurance(sentence, 'hello', 2)
>>> Start Index: 30, End Index: 35
>>> index_nth_occurance(sentence, 'hello', 2)
```

```
>>> Start Index: 0, End Index: 5
```

94 Write a program to print prime numbers from 1 to 50

```python
def is_prime(number):
    flag = True
    for i in range(2, number):
        if number % i == 0:
            flag = False
            break
    if flag:
        return True
    else:
        return False

m = filter(is_prime, range(1, 50))

for item in m:
    print(item)
```

95 Write a program to sort a list which has mix of both odd and even numbers, the sorted list should have odd numbers first and then even numbers in sorted order

```python
>>> a = [3, 4, 1, 7, 2, 12, 8, 6, 9, 11]
>>> # o/p should be [1, 3, 7, 9, 11, 2, 4, 6, 8, 12]
>>> evens = [item for item in a if item % 2 == 0]
>>> odds = [item for item in a if not item % 2 == 0]
>>> evens
[4, 2, 12, 8, 6]
>>> odds
[3, 1, 7, 9, 11]
>>> evens.sort()
>>> odds.sort()
>>> sorted_list = [*odds, *evens]
>>> sorted_list
[1, 3, 7, 9, 11, 2, 4, 6, 8, 12]
```

96 Write a program to sort a list which has mix of both odd and even numbers, in the sorted list, odd numbers should be in ascending order and even numbers should be in descending order

```python
>>> a = [3, 4, 1, 7, 2, 12, 8, 6, 9, 11]
>>> # o/p should be [1, 3, 7, 9, 11, 12, 8, 6, 4, 2]
>>> evens = [item for item in a if item % 2 == 0]
>>> odds = [item for item in a if not item % 2 == 0]
>>> evens
[4, 2, 12, 8, 6]
>>> odds
[3, 1, 7, 9, 11]
>>> evens.sort(reverse=True)
>>> evens
[12, 8, 6, 4, 2]
>>> odds.sort()
>>> odds
[1, 3, 7, 9, 11]
>>> sorted_list = [*odds, *evens]
>>> sorted_list
[1, 3, 7, 9, 11, 12, 8, 6, 4, 2]
```

97 Write a program to count the number of occurrences of non-special characters in a given string

```python
>>> s = 'hello@world! welcome!!! Python$ hi how are you & where are you?'
>>> from collections import defaultdict
>>> d = defaultdict(int)
>>> for c in s:
       if c.isalnum():
          d[c] +=1
>>> d
>>> defaultdict(<class 'int'>, {'h': 5, 'e': 7, 'l': 4, 'o': 7, 'w': 4, 'r': 4, 'd': 1, 'c': 1, 'm': 1, 'P': 1, 'y': 3, 't': 1, 'n': 1, 'i': 1, 'a': 2, 'u': 2})
```

```python
>>> s = 'hello@world! welcome!!! Python$ hi how are you & where are you?'
>>> import re
>>> chrs = re.findall(r'\w', s)
>>> {c: chrs.count(c)  for c in chrs}
```

```
>>> {'h': 5, 'e': 7, 'l': 4, 'o': 7, 'w': 4, 'r': 4, 'd': 1, 'c': 1, 'm': 1, 'P':
1, 'y': 3, 't': 1, 'n': 1, 'i': 1, 'a': 2, 'u':2}
```

98 Grouping Flowers and Animals in the below list

```python
items = ['lotus-flower', 'lilly-flower', 'cat-animal', 'sunflower-flower', 'dog-
animal']

from collections import defaultdict
d = defaultdict(list)

for item in items:
    _item, group = item.split('-')
    d[group].append(_item)
>>> d
defaultdict(<class 'list'>, {'flower': ['lotus', 'lilly', 'sunflower'], 'animal':
['cat', 'dog']})
```

99 Grouping files with same extensions

```python
files = ['apple.txt', 'yahoo.pdf', 'gmail.pdf', 'google.txt', 'amazon.pdf',
'facebook.txt', 'flipkart.pdf']

dd = defaultdict(list)
for _file in files:
    filename = _file[:-4]
    extension = _file[-3:]
    dd[extension].append(filename)

defaultdict(<class 'list'>, {'txt': ['apple', 'google', 'facebook'], 'pdf':
['yahoo', 'gmail', 'amazon', 'flipkart']})
```

100 Filter only those characters except digits

```python
s = '@hello12world34welcome!123'
temp = []
for c in s:
    if not c.isdigit():
        temp.append(c)
print(''.join(temp))
```

```
>>> @helloworldwelcome!
```

```python
>>> s = '@hello12world34welcome!123'
>>> import re
>>> re.findall(r'\D', s)
['@', 'h', 'e', 'l', 'l', 'o', 'w', 'o', 'r', 'l', 'd', 'w', 'e', 'l', 'c', 'o',
'm', 'e', '!']
>>> ''.join(re.findall(r'\D', s))
'@helloworldwelcome!'
>>>
```

101 Count number of words in a sentence. ignore special characters.
```python
>>> sentence = "Hi there! How are you:) How are you doing today!"
>>> import re
>>> words = re.findall(r'\b\w+', sentence.lower())
>>> words
['Hi', 'there', 'How', 'are', 'you', 'How', 'are', 'you', 'doing', 'today']
>>> from collections import Counter
>>> c = Counter(words)
>>> c
>>> Counter({'How': 2, 'are': 2, 'you': 2, 'Hi': 1, 'there': 1, 'doing': 1,
'today': 1})
```

102 Grouping even and odd numbers
```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# o/p should be {1: [1, 3, 5, 7, 9], 0: [2, 4, 6, 8, 10]}
from collections import defaultdict
>>> d = defaultdict(list)
>>> for number in numbers:
      if number % 2 == 0:
        d[0].append(number)
      else:
        d[1].append(number)
>>> d
>>> defaultdict(<class 'list'>, {1: [1, 3, 5, 7, 9], 0: [2, 4, 6, 8, 10]})
```

103 Find all max numbers from the below list

```python
>>> numbers = [1, 2, 3, 0, 4, 3, 2, 4, 2, 1, 0, 4]
# Output should be [4, 4, 4]
>>> max_number = max(numbers)
>>> all_max_numbers = [ number for number in numbers if number == max_number]
>>> all_max_numbers
>>> [4, 4, 4]
```

104 Find all max length words from the below sentence

```python
>>> sentence = "hello world hi apple you yahoo to you"
>>> # Output should be ['hello', 'world', 'apple', 'yahoo']
>>> max_len_word = max(sentence.split(), key=len)
>>> max_len_words = [ word for word in sentence.split() if len(word) ==
len(max_len_word)]
>>> max_len_words
>>> ['hello', 'world', 'apple', 'yahoo']
```

105 Find the range from the following string

```python
>>> sentence = '0-0, 4-8, 20-20, 43-45'
>>> # Output Should be [0, 4, 5, 6, 7, 8, 20, 43, 44, 45, 46]
>>> words = s.split(',')
>>> words
>>> ['0-0', ' 4-8', ' 20-20', ' 43-45']
>>> _range = []
>>> for word in words:
      start, end = word.split('-')
        for i in range(int(start), int(end)+1):
          _range.append(i)
>>> _range
>>> [0, 4, 5, 6, 7, 8, 20, 43, 44, 45]
```

106 Can we override a static method in python?

```python
class Parent:
    @staticmethod
    def demo():
```

```
        print('Running demo in Parent')

class Child(Parent):
    @staticmethod
    def demo():
        print('Running demo in Child')

>>> c = Child()
>>> c.demo()
>>> Running demo in Child
```

107 Write a function which returns the sum of lengths of all the iterables

```python
>>> def total_length(*iterables):
...     total = 0
...     for iterable in iterables:
...             total += len(iterable)
...     return total
...
>>> total_length([1, 2, 3], (4, 5))
5
>>> total_length([1, 2, 3], (4, 5), ['apple', 'google', 'yahoo', 'gmail',
'flipkart'])
10
>>> total_length([1, 2, 3], (4, 5), ['apple', 'google', 'yahoo', 'gmail',
'flipkart'], {1, 2, 3})
13
>>> total_length([1, 2, 3], (4, 5), ['apple', 'google', 'yahoo', 'gmail',
'flipkart'], {1, 2, 3}, {'a': 1, 'b': 2})
15
```

108 Replace whitespaces with newline character in the below string

```python
>>> sentence = "Hello world welcome to python"
>>> words = re.sub(r'\s', '\n', sentence)
>>> print(words)
Hello
world
welcome
to
python
```

```
**109 Replace all vowels with "*"**
```python
>>> sentence = "hello world welcome to python"
>>> words = re.sub(r'[aeiou]', '*', sentence)
>>> print(words)
>>> h*ll* w*rld w*lc*m* t* pyth*n
```

110 Replace all occurrences of "Java" with "Python" in a file
```python
>>> with open('java.txt', 'r') as jf:
      with open('python.txt', 'a') as pf:
         for line in jf:
            new_line = re.sub('Java', 'Python', line)
            pf.write(new_line)
```

111 Maximum sum of 3 numbers and Minimum sum of 3 numbers
```python
numbers = [10, 15, 20, 25, 30, 35, 40, 15, 15]

# Sort the list in ascending order
sorted_numbers = sorted(numbers)

# Get last 3 digits
sum_max_3 = sum(sorted_numbers[-3:])

# Get first 3 digits
sum_min_3 = sum(sorted_numbers[:3])
```

112 Write a program to get the output as below
```python
# i/p is "python@#$%pool"
# o/p should be ['PYTHON', 'POOL']

>> import re
>>> sentence = "python@#$%pool"
>>> words = re.findall(r'\w+', sentence)
>>> words
['python', 'pool']
>>> _words = [ word.upper() for word in words]
```
```

```
>>> _words
['PYTHON', 'POOL']
```

113 Write a program to print all the number which are ending with 5

```python
numbers = ['1', '12', '123', '12345', '125', '905', '55', '5', '95655', '55555']
import re
>>> for number in numbers:
      match = re.findall('5$', number)
       if match:
         print(number)
12345
125
905
55
5
95655
55555
>>>
```

114 Write a program to get the indexes of each item in the below list

```python
names = ['apple', 'google', 'apple', 'yahoo', 'yahoo', 'google', 'gmail', 'gmail',
'gmail']
output should be -  {'apple': [0, 2], 'google': [1, 5], 'yahoo': [3, 4], 'gmail':
[6, 7, 8]}

from collections import defaultdict
item_index = defaultdict(list)

for index, name in enumerate(names):
    item_index[name].append(index)

>>> item_index
defaultdict(<class 'list'>, {'apple': [0, 2], 'google': [1, 5], 'yahoo': [3, 4],
'gmail': [6, 7, 8]})
```

115 Write a program to print "Bangalore" 10 times without using "for" loop

```python
>>> print("Bangalore \n" * 10)
Bangalore
```

```
Bangalore
Bangalore
Bangalore
Bangalore
Bangalore
Bangalore
Bangalore
Bangalore
Bangalore
```

116 Write a program to print all the words which starts with letter 'h' in the given string

```python
>>> import re
>>>
>>> matches = re.findall(r"\bh\w+", 'hello world hi hello universe how are you happy birthday')
>>>
>>> matches
['hello', 'hi', 'hello', 'how', 'happy']
```

117 Write a program to sum all even numbers in the given string

```python
>>> sentence = 'hello 123 world 567 welcome to 9724 python'
>>> import re
>>> numbers = re.findall(r'\d', sentence)
>>> numbers
['1', '2', '3', '5', '6', '7', '9', '7', '2', '4']
>>> # Typecast each item of the list to int
>>> numbers = [ int(number) for number in numbers]
>>> numbers
[1, 2, 3, 5, 6, 7, 9, 7, 2, 4]
>>> # Filter out even numbers
>>> even_numbers = [ number  for number in numbers if number % 2 == 0]
>>> even_numbers
[2, 6, 2, 4]
>>> # Get the sum
>>> sum(even_numbers)
14
```

118 Write a program to add each number in word1 to number in word2

```python
# e.g. 1 + 5, 2 + 6, 3 + 7, 4 + 8, 5 + 9
>>> word1 = 'hello 1 2 3 4 5'
>>> word2 = 'world 5 6 7 8 9'
>>> numbers1 = re.findall(r'\d', word1)
>>> numbers2 = re.findall(r'\d', word2)
>>> total = [ ]
>>> for n1, n2 in zip(numbers1, numbers2):
       total.append(int(n1) + int(n2))
```

119 Write a program to filter out even and odd numbers in the given string

```python
>>> sentence = 'hello 123 world 456 welcome to python498675634'
>>> import re
>>> numbers = re.findall(r'\d', sentence)
>>> numbers
['1', '2', '3', '4', '5', '6', '4', '9', '8', '6', '7', '5', '6', '3', '4']
>>> even_numbers = [  item for item in numbers if int(item) % 2 == 0]
>>> even_numbers
['2', '4', '6', '4', '8', '6', '6', '4']
>>> odd_numbers = [  item for item in numbers if int(item) % 2 == 1]
>>> odd_numbers
['1', '3', '5', '9', '7', '5', '3']
```

120 Write a program to print all the number which are starting with 8

```python
>>> numbers = ['857', '987', '8', '120', '888888', '547', '7674', '89', '589',
'388888', '2889']
>>> import re
>>> for number in numbers:
       match = re.findall(r'^8', number)
       if match:
          print(number)
857
8
888888
89
```

121 Write a program to remove duplicates from the list without using set or empty list

```python
>>> l1 = [1, 2, 3, 4, 1, 2, 3, 4, 3, 4, 4]
>>> # Make copy of original list
>>> l2 = l1[::] # or l2 = l1.copy()
>>> l2
[1, 2, 3, 4, 1, 2, 3, 4, 3, 4, 4]
>>> for item in l2:
...     if l1.count(item) > 1:
...             l1.remove(item)
...
>>> l1
[1, 2, 3, 4]
```

122 Print all the missing numbers from 1 to 10 in the below list

```python
>>> numbers = [1, 3, 6, 8, 10]
>>> for i in range(1, 11):
...     if not i in l:
...             print(f'missing number is {i}')
missing number is 2
missing number is 4
missing number is 5
missing number is 7
missing number is 9
```

123 Write a python program to get the below output

```python
>>> l1 = [1, 2, 3]
>>> l2 = ['a', 'b', 'c']
>>> # o/p ['1a', '1b', '1c', '2a', '2b', '2c', '3a', '3b', '3c']
>>> # Convert x to str.
>>> result = [ ''.join((str(x), y)) for x in l1 for y in l2]
>>> result
['1a', '1b', '1c', '2a', '2b', '2c', '3a', '3b', '3c']
```

124 Write a python program to get the below output

```python
>>> word = "AAAAaaccYYY"
>>> # o/p ['Y3', 'c2', 'A4', 'a2']

>>> # Get all the unique characters
>>> unique_letters = set(word)

>>> # build a list of character and its count pair
>>> count = [ ''.join((letter, str(word.count(letter)))) for letter in
unique_letters ]
>>> count
['Y3', 'c2', 'A4', 'a2']
```

125 What is the output of the below function call
```python
class Demo:
    def greet(self):
        print("hello world")

    def greet(self):
        print("hello universe")

>>> d = Demo()
>>> d.greet()
hello universe
>>>
```

126 In the list below, find all the number pairs which results in 10 either when added or subtracted
```python
>>> a = [3, 5, -4, 8, 11, 1, -1, 6]

>>> for item1 in a:
>>>    for item2 in a:
>>>        if item1 != item2:
>>>            if item1 + item2 == 10 or item1 - item2 == 10:
>>>                print((item1, item2))

>>> (11, 1)
>>> (11, -1)
>>> (-1, 11)
```

```
>>> (6, -4)
>>>
```

127 Write a decorator to prefix +91 to the original phone numbers

```python
numbers = [1234567890, 123456790, 1234567890]

def prefix_country_code(func):
    def wrapper(*args, **kwargs):
        numbers, = args
        prefix_numbers = [ "+91-"+str(number) for number in numbers ]
        return func(prefix_numbers)
    return wrapper

@prefix_country_code
def print_numbers(numbers):
    for number in numbers:
        print(number)
```

128 Write a program to get the below output

```python
>>> d = {"a": 1, "b": 2, "c": 3, "d": 4, "e": 5}
>>> # o/p should be ['b', 'd']

>>> # First get all the keys into a list
>>> keys = list(d.keys())
>>> result = [ key for key in keys[1::2] ]
>>> result
>>> ['b', 'd']
```

129 Can we have multiple __init__ methods in a class

```python
class Point:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __init__(self, a, b, c):
        self.a = a
        self.b = b
```

```
        self.c = c

>>> p = Point(1, 2, 3)
>>> p.a
1
>>> p.b
2
>>> p.c
3
>>> p = Point(1, 2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Point.__init__() missing 1 required positional argument: 'c'
>>>

# Since Python is dynamically typed language, it takes the latest __init__ method.
```

130 Why python is Object Oriented

```python
       Python is object oriented because everyting in python is an object (defined
as class)
```

131 What are .pyc files

```python
1. pyc files are python compiled.
2. Once .py file is compiled by python compiler, it generates .pyc file.
3. .pyc files contains byte code which is understandable by python virtual machine.
4. pyc files are generated when a python module is imported.
5. Python compiler will not compile a python module again and again unless there is
a change in code.
6. This makes programs to run faster since byte code for a module is already
generated.
```

132 Reverse a list without using any built-in functions and slicing

```python
>>> a = [1, 2, 3, 4, 5]
>>> ra = [ a[index]  for index in range(len(a)-1, -1, -1) ]
>>> ra
[5, 4, 3, 2, 1]
```

133 Write a program to get the below output

```python
>>> # i/p = "10.20.30.40"
>>> # o/p = "40.30.20.10"
>>> ip = "10.20.30.40"
>>> parts = ip.split(".")
>>> ".".join(parts[::-1])
>>> parts
>>> '40.30.20.10'
```