Exercises - Python Functions :

1.Write a Python program to print the following string in a specific format (see the output).

Sample String: "Twinkle, twinkle, little star, How I wonder what you are! Up above the world so high, Like a diamond in the sky. Twinkle, twinkle, little star, How I wonder what you are!"

```
print("Twinkle, twinkle, little star, \n\tHow I wonder what you are! \n\t\tUp
above the world so high, \n\t\tLike a diamond in the sky. \nTwinkle, twinkle,
little star, \n\tHow I wonder what you are!")
```

2. Write a Python program to get the Python version you are using

```
import sys
print("Python version")
print (sys.version)
print("Version info.")
print (sys.version_info)
```

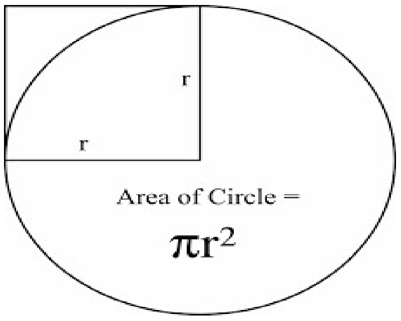
3. Write a Python program to display the current date and time.

```
import datetime
now = datetime.datetime.now()
print ("Current date and time : ")
print (now.strftime("%Y-%m-%d %H:%M:%S"))
```

4. Write a Python program which accepts the radius of a circle from the user and compute the area

Given , the radius , r = 1.1 CM

In geometry, the area enclosed by a circle of radius r is πr2. Here the Greek letter π represents a constant, approximately equal to 3.14159, which is equal to the ratio of the circumference of any circle to its diameter.



```
from math import pi
r = float(input ("Input the radius of the circle : "))
print ("The area of the circle with radius " + str(r) + " is: " +
str(pi * r**2))
```

5. Write a Python program which accepts the user's first and last name and print them in reverse order with a space between them

```
fname = input("Input your First Name : ")
lname = input("Input your Last Name : ")
print ("Hello  " + lname + " " + fname)
```

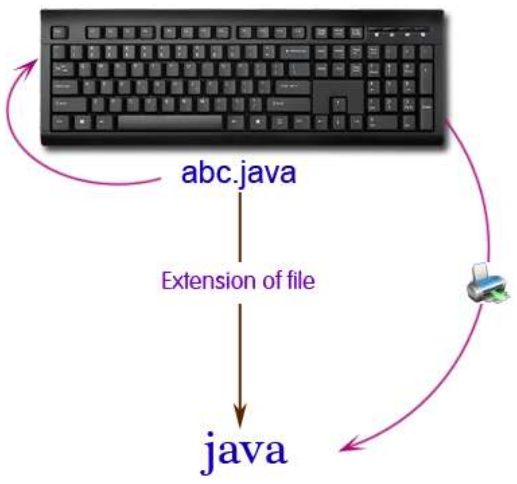
James Bond -→ Bond James

6. Write a Python program which accepts a sequence of comma-separated numbers from user and generate a list and a tuple with those numbers

Sample data : 3, 5, 7, 23

```python
values = input("Input some comma seprated numbers : ")
list = values.split(",")
tuple = tuple(list)
print('List : ',list)
print('Tuple : ',tuple)
```

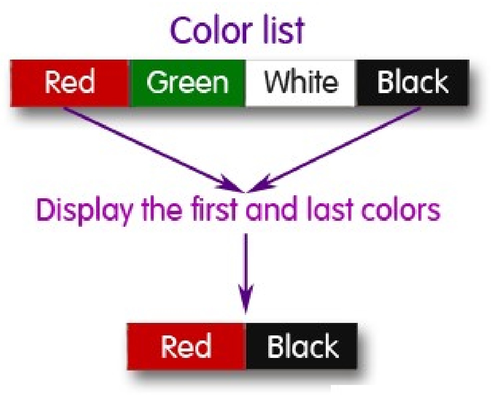
7. Write a Python program to accept a filename from the user and print the extension of that.



```python
filename = input("Input the Filename: ")
f_extns = filename.split(".")
print ("The extension of the file is : " + repr(f_extns[-1]))
```

8. Write a Python program to display the first and last colors from the following list.

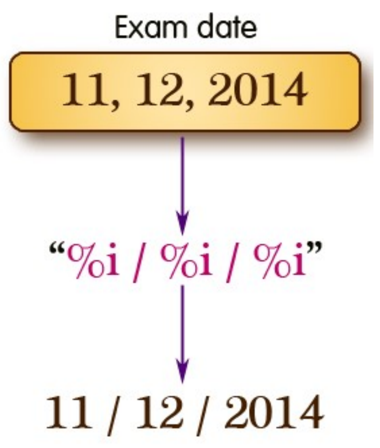
color_list = ["Red","Green","White" ,"Black"]



```
color_list = ["Red","Green","White" ,"Black"]
print( "%s %s"%(color_list[0],color_list[-1]))
```

9. Write a Python program to display the examination schedule. (extract the date from exam_st_date).

exam_st_date = (11, 12, 2014)



```
exam_st_date = (11,12,2014)
print( "The examination will start from : %i / %i / %i"%exam_st_date)
```

#10. Write a Python program that accepts an integer (n) and computes the value **of n+nn+nnn**

Sample value of n is 5
Expected Result : 615

```
a = int(input("Input an integer : "))
n1 = int( "%s" % a )
n2 = int( "%s%s" % (a,a) )
n3 = int( "%s%s%s" % (a,a,a) )
print (n1+n2+n3)
```

~~**11.** Write a Python program to print the documents (syntax, description etc.) of Python built-in function(s).~~
~~*Sample function* : abs()~~
~~*Expected Result* :~~
~~abs(number) -> number~~
~~Return the absolute value of the argument.~~

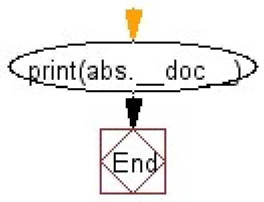
~~print(abs.__doc__)~~

~~*Sample function*: abs()~~

~~**Python Docstring:**~~

~~A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. Such a docstring becomes the __doc__ special attribute of that object.~~

~~All modules should normally have docstrings, and all functions and classes exported by a module should also have docstrings. Public methods (including the __init__ constructor) should also have docstrings.~~



12. Write a Python program to print the calendar of a given month and year.
Note : Use 'calendar' module.

```
import calendar
y = int(input("Input the year : "))
m = int(input("Input the month : "))
print(calendar.month(y, m))
```

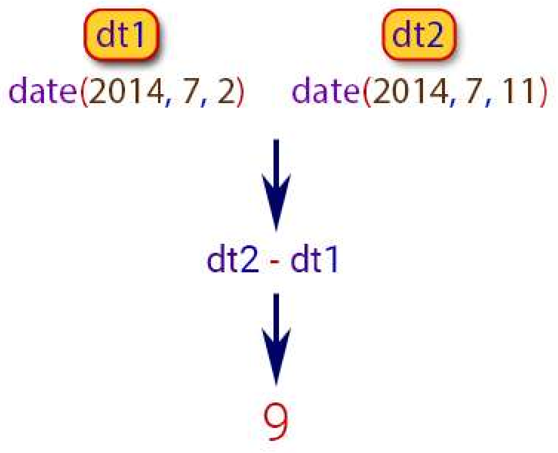
13. Write a Python program to print the following here document.

Sample string :

```
a string that you "don't" have to escape
This
is a ....... multi-line
heredoc string --------> example
```

```python
print("""
a string that you "don't" have to escape
This
is a  ....... multi-line
heredoc string --------> example
""")
```

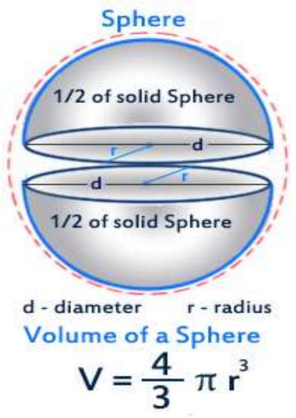
14. Write a Python program to calculate number of days between two dates.
Sample dates : (2014, 7, 2), (2014, 7, 11)
Expected output : 9 days



```python
from datetime import date
f_date = date(2014, 7, 2)
l_date = date(2014, 7, 11)
delta = l_date - f_date
print(delta.days)
```

15. Write a Python program to get the volume of a sphere with **radius 6.**

The volume of the sphere is : V = 4/3 × π × r3 = π × d3/6.



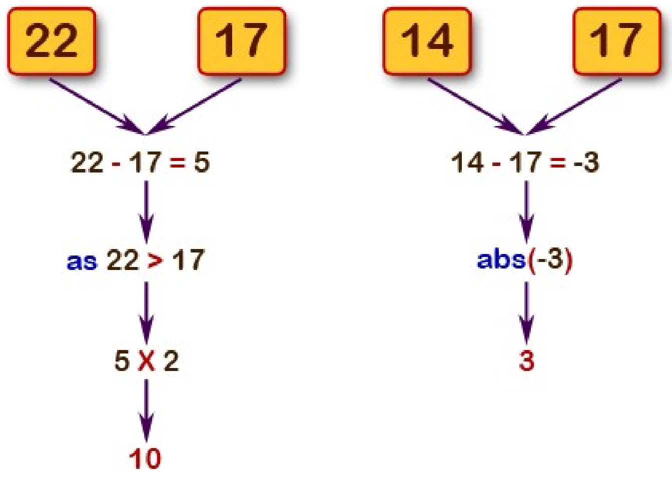
$$V = \frac{4}{3}\pi r^3$$

```
pi = 3.1415926535897931
r= 6.0
V= 4.0/3.0*pi* r**3
print('The volume of the sphere is: ',V)
```

16. Write a Python program to get the difference between a given number and 17, if the number is greater than 17 return double the absolute difference.

Python if-else syntax:

```
if condition :
indentedStatementBlockForTrueCondition
else:
indentedStatementBlockForFalseCondition
```

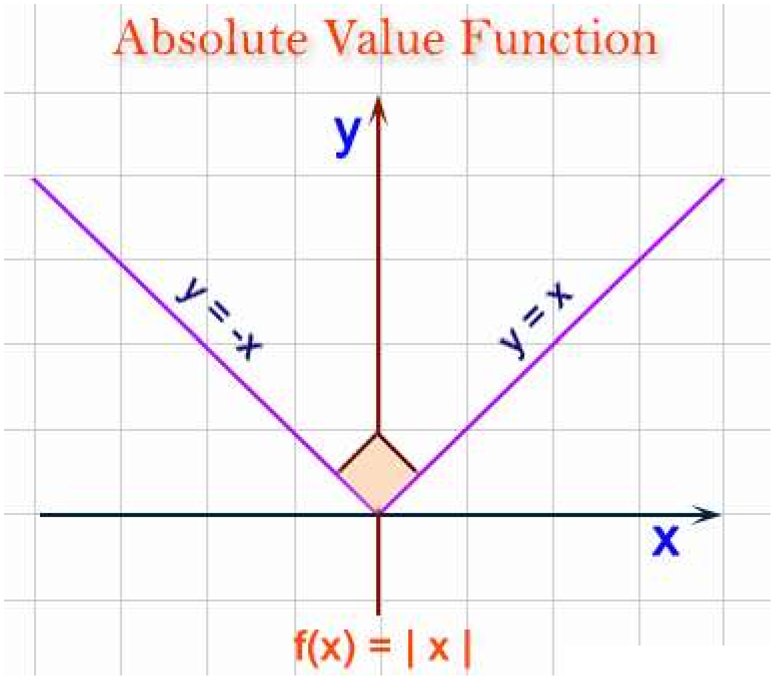


```
def difference(n):
    if n <= 17:
        return 17 - n
    else:
        return (n - 17) * 2

print(difference(22))
print(difference(14))
```

17. Write a Python program to test whether a number is within 100 of 1000 or 2000

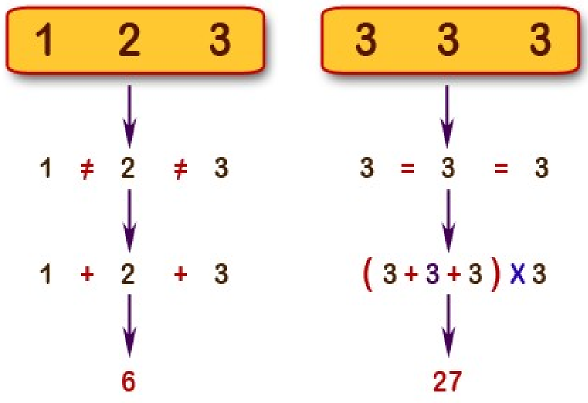
Python abs(x) function:

The function returns the absolute value of a number. The argument may be an integer or a floating point number. If the argument is a complex number, its magnitude is returned.

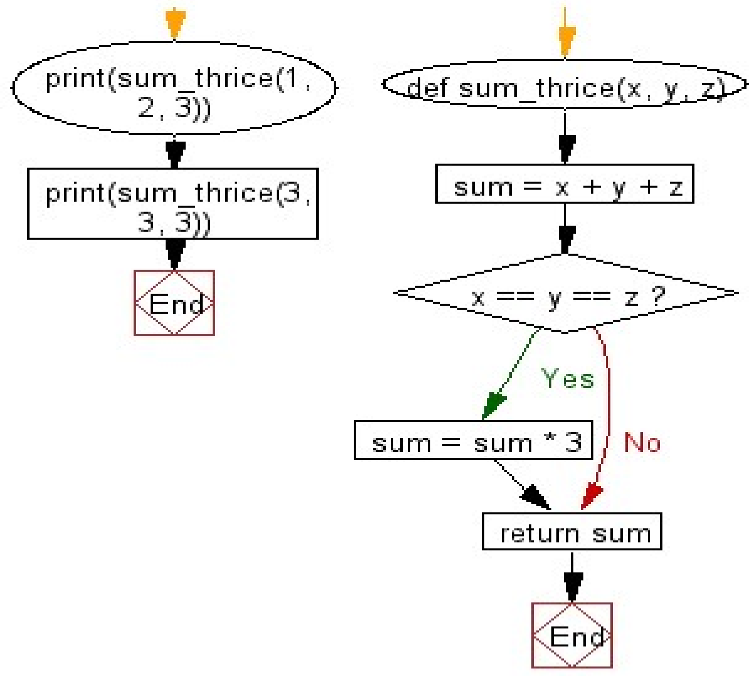


```
def near_thousand(n):
      return ((abs(1000 - n) <= 100) or (abs(2000 - n) <= 100))
print(near_thousand(1000))
print(near_thousand(900))
print(near_thousand(800))
print(near_thousand(2200))
```

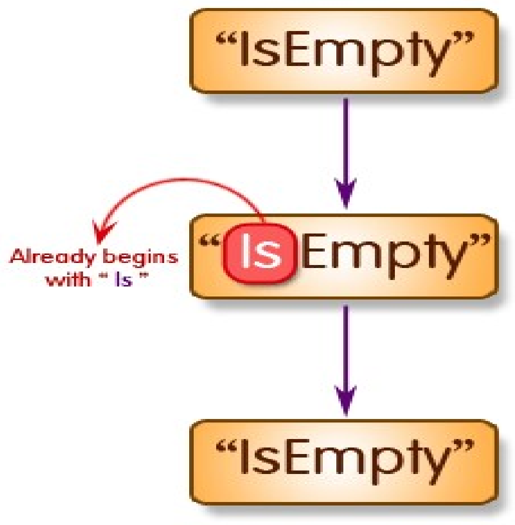
18. Write a Python program to calculate the sum of three given numbers, if the values are equal then return three times of their sum



```
def sum_thrice(x, y, z):

     sum = x + y + z

     if x == y == z:
      sum = sum * 3
     return sum

print(sum_thrice(1, 2, 3))
print(sum_thrice(3, 3, 3))
```

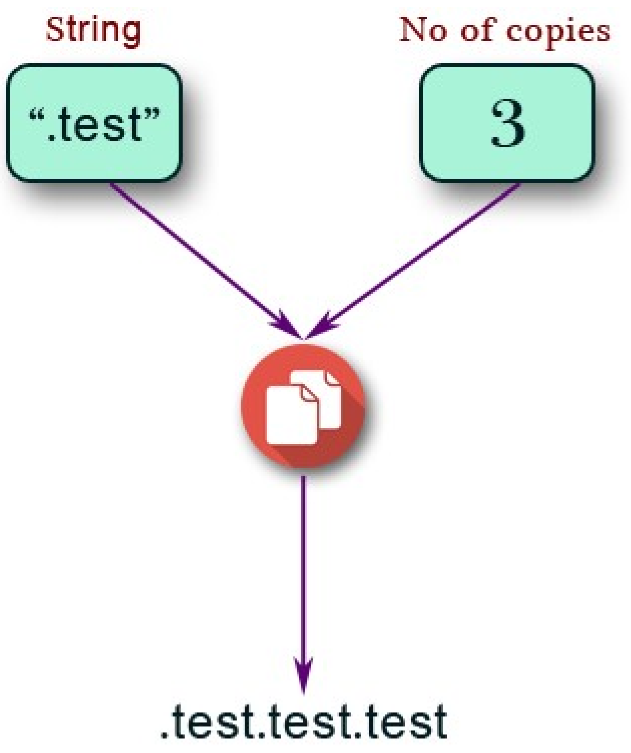


19. Write a Python program to get a new string from a given string where "Is" has been added to the front. If the given string already begins with "Is" then return the string unchanged.



```
def new_string(str):
  if len(str) >= 2 and str[:2] == "Is":
    return str
  return "Is" + str

print(new_string("Array"))
print(new_string("IsEmpty"))
```

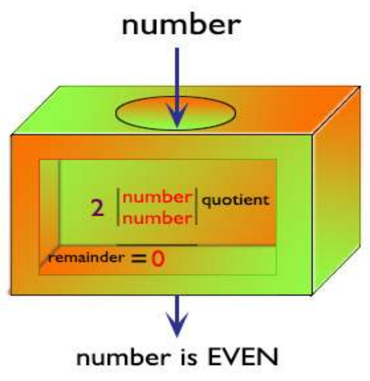
20. Write a Python program to get a string which is n (non-negative integer) copies of a given string.



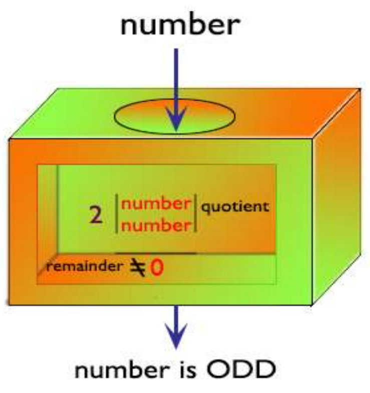
```
def larger_string(str, n):
    result = ""
    for i in range(n):
        result = result + str
    return result

print(larger_string('abc', 2))
print(larger_string('.py', 3))
```

21. Write a Python program to find whether a given number (accept from the user) is even or odd, print out an appropriate message to the user

Even Numbers

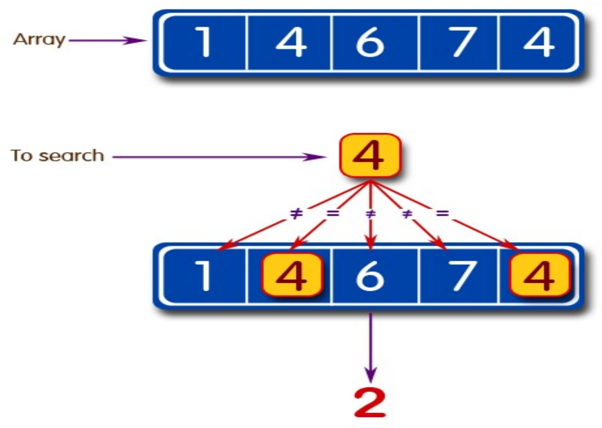


Odd Numbers



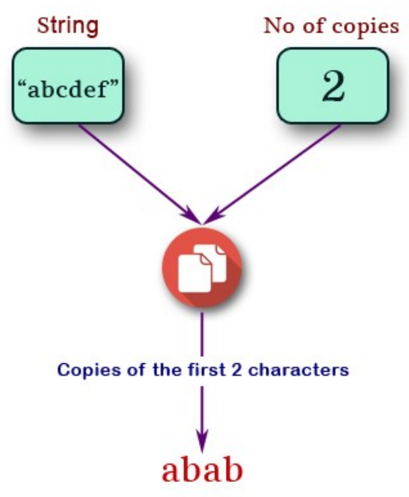
```
num = int(input("Enter a number: "))

mod = num % 2

if mod > 0:

    print("This is an odd number.")

else:

    print("This is an even number.")
```

22. Write a Python program to count the number 4 in a given list



```
def list_count_4(nums):
  count = 0
  for num in nums:
    if num == 4:
      count = count + 1

  return count

print(list_count_4([1, 4, 6, 7, 4]))
print(list_count_4([1, 4, 6, 4, 7, 4]))
```

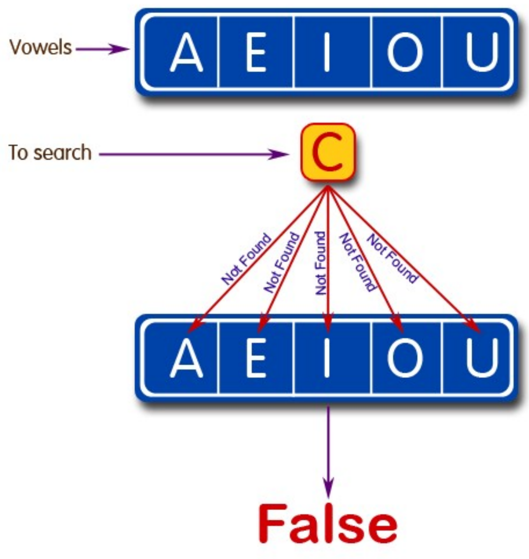
23. Write a Python program to get the n (non-negative integer) copies of the first 2 characters of a given string. Return the n copies of the whole string if the length is less than 2.



```
def substring_copy(str, n):
  flen = 2
  if flen > len(str):
    flen = len(str)
  substr = str[:flen]

  result = ""
  for i in range(n):
    result = result + substr
  return result
print(substring_copy('abcdef', 2))
print(substring_copy('p', 3));
```

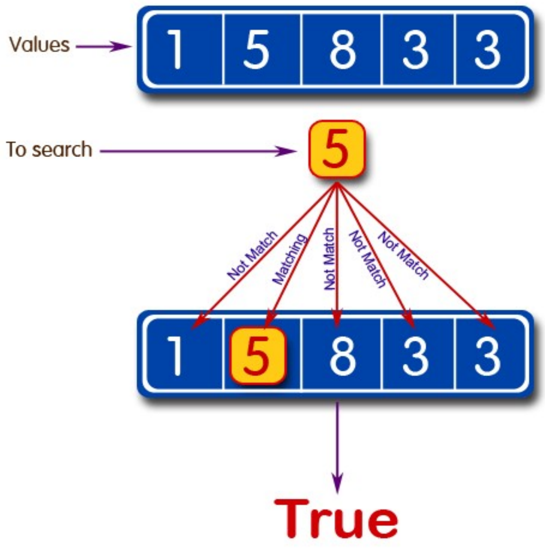
24. Write a Python program to test whether a passed letter is a vowel or not

```
def is_vowel(char):
    all_vowels = 'aeiou'
    return char in all_vowels
print(is_vowel('c'))
print(is_vowel('e'))
```



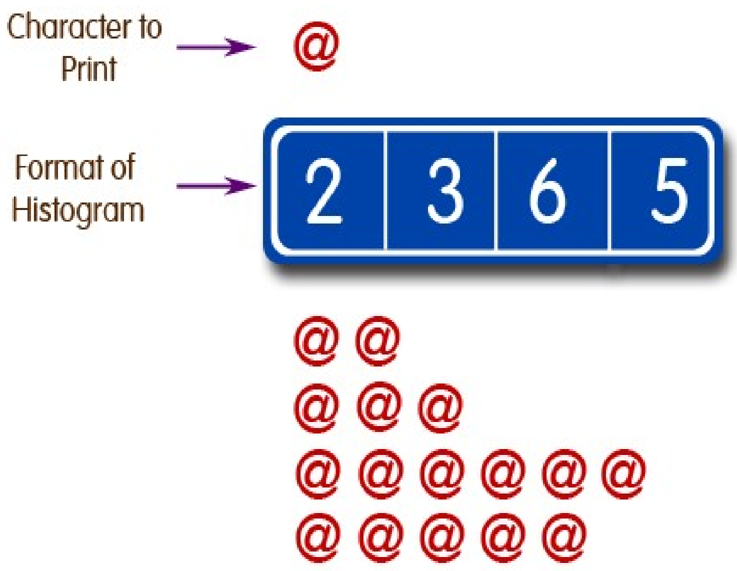
25. Write a Python program to check whether a specified value is contained in a group of values

Test Data :
3 -> [1, 5, 8, 3] : True
-1 -> [1, 5, 8, 3] : False



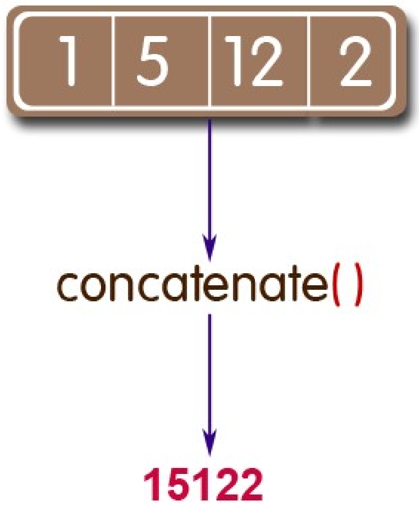
26. Write a Python program to create a histogram from a given list of integers

```
def histogram( items ):
    for n in items:
        output = ''
        times = n
        while( times > 0 ):
          output += '*'
          times = times - 1
        print(output)

histogram([2, 3, 6, 5])
```



27. Write a Python program to concatenate all elements in a list into a string and return it.

```
def concatenate_list_data(list):
    result= ''
    for element in list:
        result += str(element)
    return result

print(concatenate_list_data([1, 5, 12, 2]))
```



28. Write a Python program to print all even numbers from a given numbers list in the same order and stop the printing if any numbers that come after 237 in the sequence.

Sample numbers list :

numbers = [

386, 462, 47, 418, 907, 344, 236, 375, 823, 566, 597, 978, 328, 615, 953, 345,

399, 162, 758, 219, 918, **237**, 412, 566, 826, 248, 866, 950, 626, 949, 687, 217,

815, 67, 104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831, 445, 742, 717,

958,743, 527

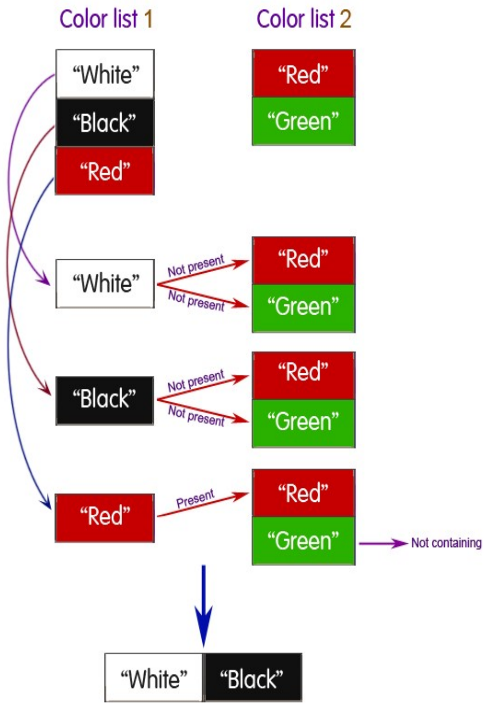
]

```
#Code:
```

```
numbers = [
    386, 462, 47, 418, 907, 344, 236, 375, 823, 566, 597, 978, 328,
615, 953, 345,
    399, 162, 758, 219, 918, 237, 412, 566, 826, 248, 866, 950, 626,
949, 687, 217,
    815, 67, 104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831,
445, 742, 717,
    958,743, 527
    ]

for x in numbers:
    if x == 237:
        print(x)
        break;
    elif x % 2 == 0:
        print(x)
```

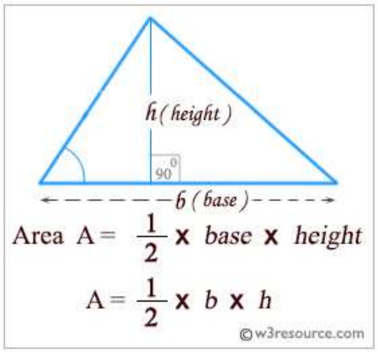
29. Write a Python program to print out a set containing all the colors from color_list_1 which are not present in color_list_2.

Tet Data :
color_list_1 = set(["White", "Black", "Red"])
color_list_2 = set(["Red", "Green"])
Expected Output :
{'Black', 'White'}



```
color_list_1 = set(["White", "Black", "Red"])
color_list_2 = set(["Red", "Green"])

print(color_list_1.difference(color_list_2))
```

30. Write a Python program that will accept the base and height of a triangle and compute the area

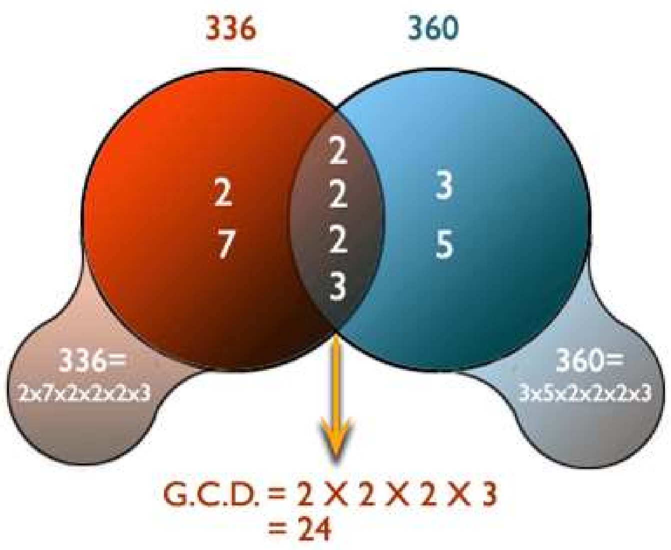

h (height)
90°
b (base)
Area A = $\frac{1}{2}$ x base x height
A = $\frac{1}{2}$ x b x h
©w3resource.com

- Vertex of a triangle: The point at which two sides of a triangle meet.
- Altitude of a triangle: The perpendicular segment from

  a vertex of a triangle to the line containing the opposite side.
- Base of a triangle: The side of a triangle to which an altitude is drawn.
- Height of a triangle: The length of an altitude.

```
b = int(input("Input the base : "))
h = int(input("Input the height : "))

area = b*h/2

print("area = ", area)
```

31. Write a Python program to compute the greatest common divisor (GCD) of two positive integers.



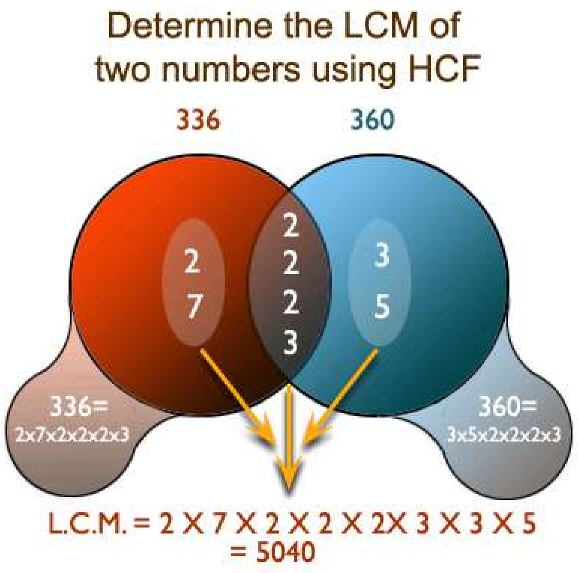
```
def gcd(x, y):
    gcd = 1

    if x % y == 0:
        return y
```

```
 for k in range(int(y / 2), 0, -1):
        if x % k == 0 and y % k == 0:
            gcd = k
            break
    return gcd

print(gcd(12, 17))
     print(gcd(4, 6))
```

OUTPUT:

```
     1
     2
```

32. Write a Python program to get the least common multiple (LCM) of two positive integers.


Determine the LCM of two numbers using HCF
336
360
2
7
2
2
2
3
3
5
336= 2x7x2x2x2x3
360= 3x5x2x2x2x3
L.C.M. = 2 X 7 X 2 X 2 X 2X 3 X 3 X 5
= 5040

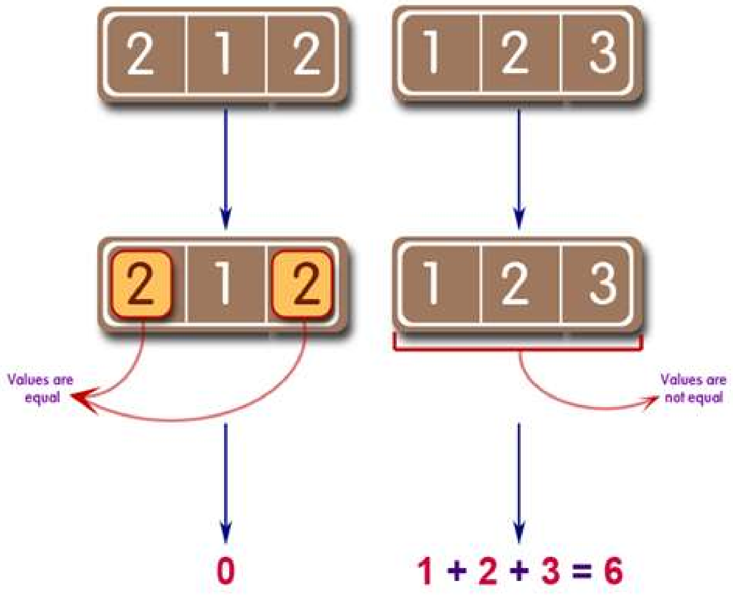
```
def lcm(x, y):
   if x > y:
       z = x
   else:
       z = y

   while(True):
       if((z % x == 0) and (z % y == 0)):
           lcm = z
```

```
              break
        z += 1

   return lcm
print(lcm(4, 6))
     print(lcm(15, 17))
```

OUTPUT:

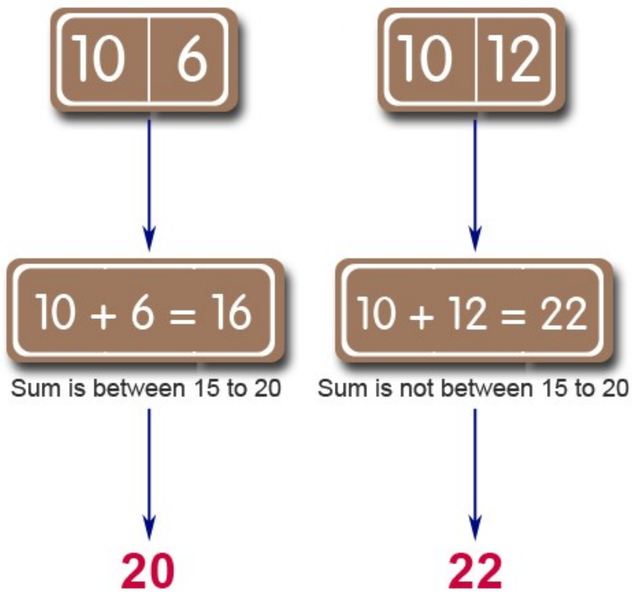
```
12
255
```

33. Write a Python program to sum of three given integers. However, if two values are equal sum will be Zero



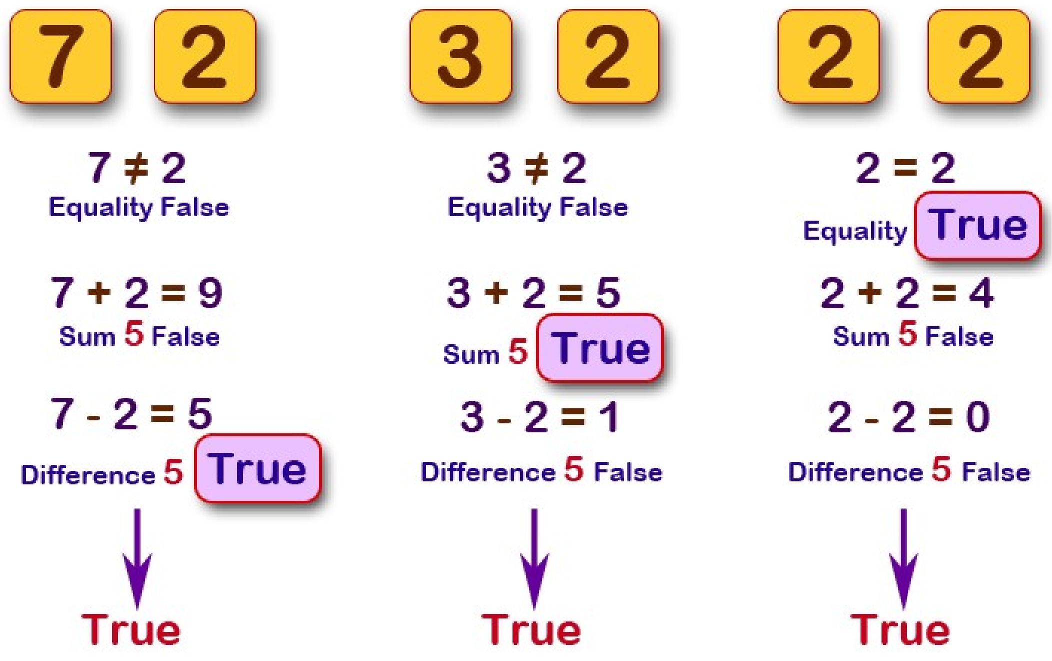
```
def sum(x, y, z):
    if x == y or y == z or x==z:
        sum = 0
    else:
        sum = x + y + z
    return sum

print(sum(2, 1, 2))
print(sum(3, 2, 2))
print(sum(2, 2, 2))
print(sum(1, 2, 3))
```

34. Write a Python program to sum of two given integers. However, if the sum is between 15 to 20 it will return 20



```
def sum(x, y):
    sum = x + y
    if sum in range(15, 20):
        return 20
    else:
        return sum

print(sum(10, 6))
print(sum(10, 2))
print(sum(10, 12))
```

35. Write a Python program that will return **true** if the two given integer values are equal or their sum or difference is 5.



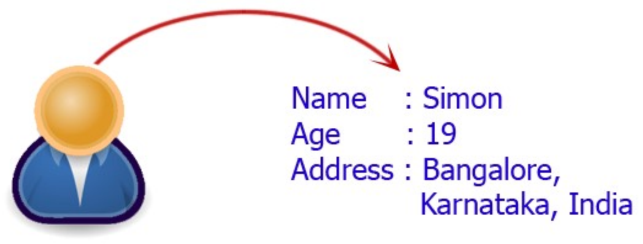
```python
def test_number5(x, y):
    if x == y or abs(x-y) == 5 or (x+y) == 5:
        return True
    else:
        return False


print(test_number5(7, 2))
print(test_number5(3, 2))
print(test_number5(2, 2))
print(test_number5(9, 2))
```

36. Write a Python program to add two objects if both objects are an integer type

```
def add_numbers(a, b):
    if not (isinstance(a, int) and isinstance(b, int)):
        raise TypeError("Inputs must be integers")
    return a + b

print(add_numbers(10, 20))
```

Output : 30

37. Write a Python program to display your details like name, age, address in three different lines



```
def personal_details():
    name, age = "Simon", 19
    address = "Bangalore, Karnataka, India"
    print("Name: {}\nAge: {}\nAddress: {}".format(name, age, address))

personal_details()
```

38. Write a Python program to solve (x + y) * (x + y)
Test Data : x = 4, y = 3
Expected Output : (4 + 3) ^ 2) = 49

(a+b)**2

BODMAS =

```
x, y = 4, 3
result = x * x + 2 * x * y + y * y
print("({} + {}) ^ 2) = {}".format(x, y, result))
```

Output :

(4 + 3) ^ 2) = 49

39. Write a Python program to compute the future value of a specified principal amount, rate of interest, and a number of years.

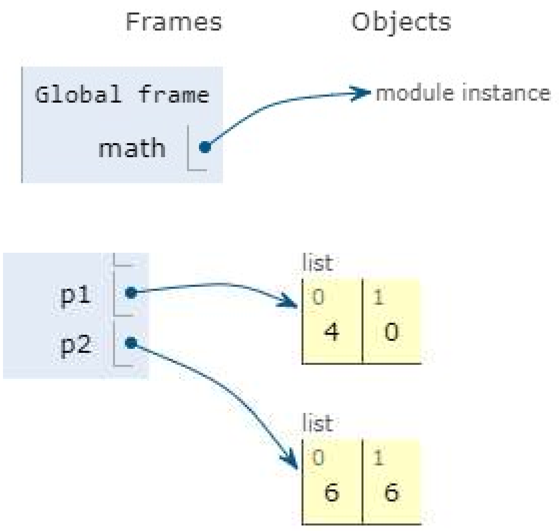
```
amt = 5000
int = 3.5
years = 7

future_value  = amt*((1+(0.01*int)) ** years)
print(round(future_value,2))
```

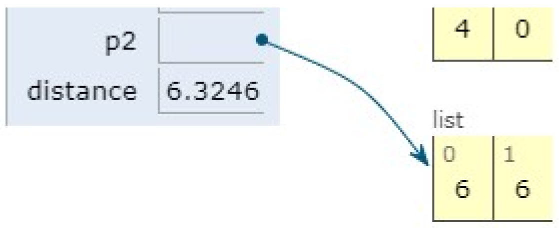
Output;

12722.79

40. Write a Python program to compute the distance between the points (x1, y1) and (x2, y2)

```
import math
p1 = [4, 0]
p2 = [6, 6]
distance = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )

print(distance)
```





41. Write a Python program to check whether a file exists

```
import os.path
open('abc.txt', 'w')
print(os.path.isfile('abc.txt'))
```

42. Write a Python program to determine if a Python shell is executing in 32bit or 64bit mode on OS

```
# For 32 bit it will return 32 and for 64 bit it will return 64
import struct
print(struct.calcsize("P") * 8)
```

Output

64

43. Write a Python program to get OS name, platform and release information

```
import platform

import os

print(os.name)

print(platform.system())

print(platform.release())
```

Output:

```
nt
Windows
10
```

44. Write a Python program to locate Python site-packages

```
import site;
print(site.getsitepackages())
```

Output

```
'c:\\users\\xsr89\\appdata\\local\\programs\\python\\python37-32', 'c:\\us
ers\\xsr89\\appdata\\local\\programs\\python\\python37-32\\lib\\site-packa
ges'
```

~~**45.** Write a python program to call an external command in Python~~

```
~~from subprocess import call~~
~~call(["ls", "-l"])~~
```

~~46. Write a python program to get the path and name of the file that is currently executing.~~

47. Write a Python program to find out the number of CPUs using.

```
import multiprocessing
print(multiprocessing.cpu_count())
```

Output

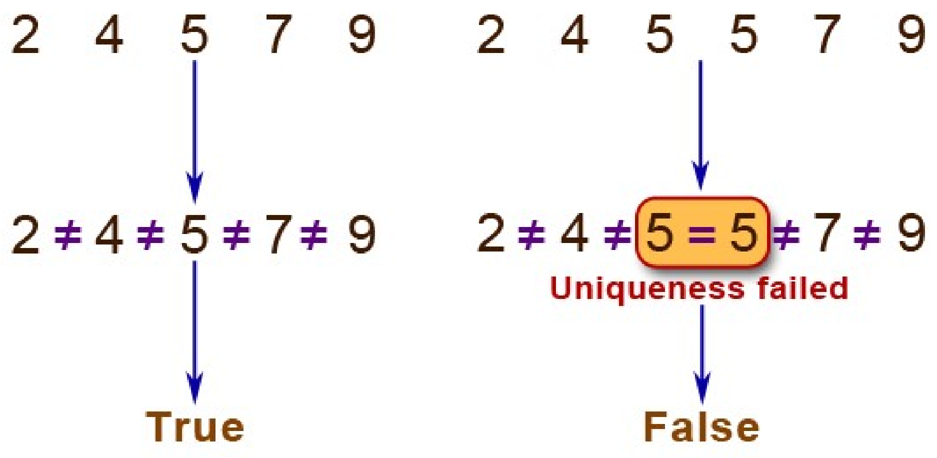
4

~~48. Write a Python program to parse a string to Float or Integer.~~

- 49 Write a Python program to get the details of math module
- Write a Python program to calculate midpoints of a line.
- Write a Python program to hash a word.
- Write a Python program to get the copyright information.
- Write a Python program to get the command-line arguments (name of the script, the number of arguments, arguments) passed to a scrip
-

Part – 2 :

1. Write a Python function that takes a sequence of numbers and determines if all the numbers are different from each other.

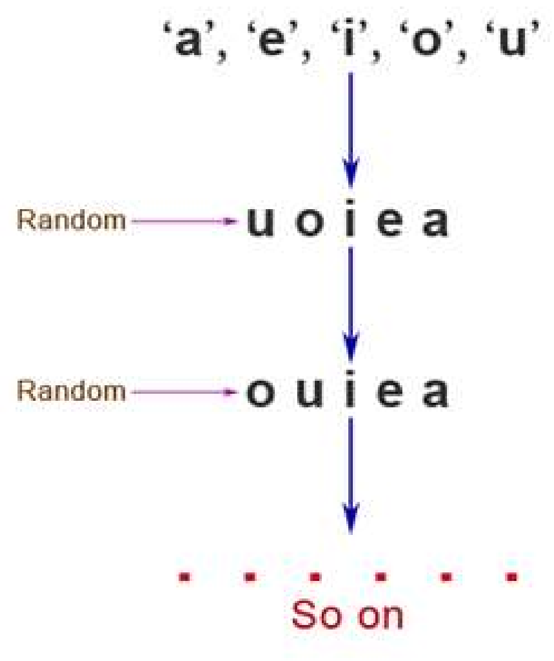


```
def test_distinct(data):
  if len(data) == len(set(data)):
    return True
  else:
    return False;
print(test_distinct([1,5,7,9]))
print(test_distinct([2,4,5,5,7,9]))
```

Output

```
True

False
```

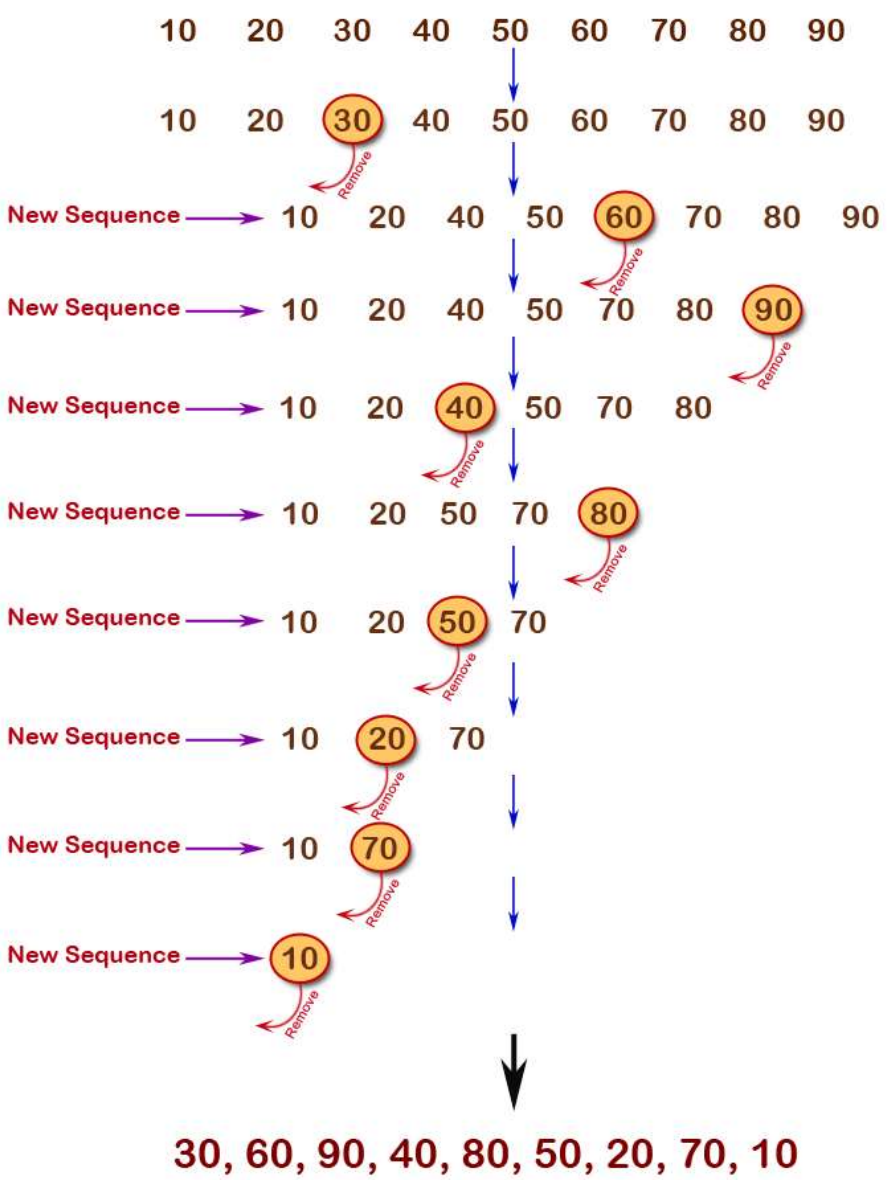
2. Write a Python program to create all possible strings by using 'a', 'e', 'i', 'o', 'u'. Use the characters exactly once



```
import random
char_list = ['a','e','i','o','u']
random.shuffle(char_list)
print(''.join(char_list))
```

```
iauoe
```

3. Write a Python program to remove and print every third number from a list of numbers until the list becomes empty.

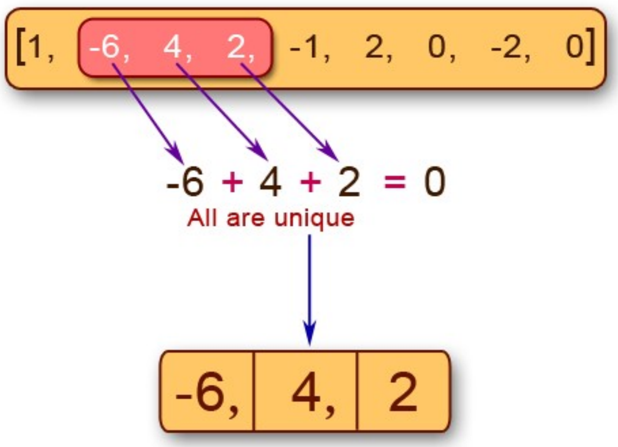


```
def remove_nums(int_list):

  #list starts with 0 index

  position = 3 - 1

  idx = 0

  len_list = (len(int_list))
  while len_list>0:

    idx = (position+idx)%len_list

    print(int_list.pop(idx))

    len_list -= 1

nums = [10,20,30,40,50,60,70,80,90]

remove_nums(nums)
```

Output

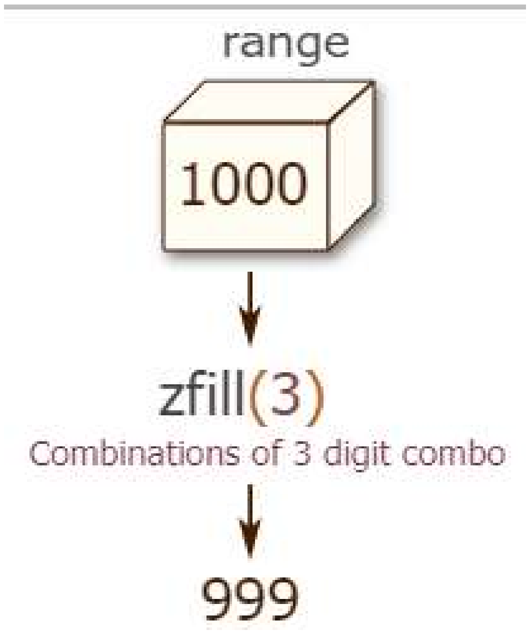
```
30
60
90
40
80
50
20
70
10
```

4. Write a Python program to find unique triplets whose three elements gives the sum of zero from an array of n integers.



```
def three_sum(nums):
  result = []
  nums.sort()
  for i in range(len(nums)-2):
    if i> 0 and nums[i] == nums[i-1]:
      continue
    l, r = i+1, len(nums)-1
    while l < r:
      s = nums[i] + nums[l] + nums[r]
      if s > 0:
        r -= 1
      elif s < 0:
          l += 1
      else:
        # found three sum
        result.append((nums[i], nums[l], nums[r]))
        # remove duplicates
        while l < r and nums[l] == nums[l+1]:
          l+=1
          while l < r and nums[r] == nums[r-1]:
            r -= 1
            l += 1
            r -= 1
          return result

x = [1, -6, 4, 2, -1, 2, 0, -2, 0 ]
print(three_sum(x))
```

5. Write a Python program to create the combinations of 3 digit combo



```
numbers = []
for num in range(1000):
  num=str(num).zfill(3)
print(num)
numbers.append(num)
```

6. Write a Python program to print a long text, convert the string to a list and print all the words and their frequencies.

```
string_words = '''United States Declaration of Independence
From Wikipedia, the free encyclopedia
The United States Declaration of Independence is the statement
adopted by the Second Continental Congress meeting at the Pennsylvania
State
House (Independence Hall) in Philadelphia on July 4, 1776, which
announced
that the thirteen American colonies, then at war with the Kingdom of
Great
Britain, regarded themselves as thirteen independent sovereign states,
no longer
```

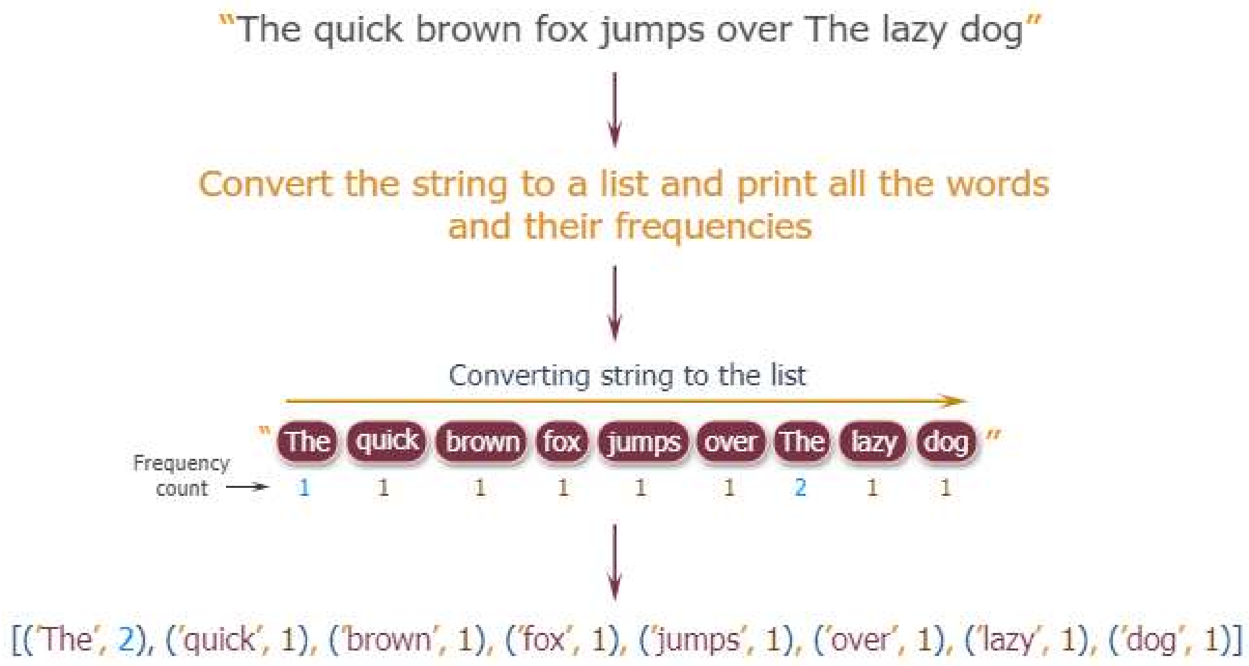
under British rule. These states would found a new nation – the United States of
America. John Adams was a leader in pushing for independence, which was passed
on July 2 with no opposing vote cast. A committee of five had already drafted the
formal declaration, to be ready when Congress voted on independence.

John Adams persuaded the committee to select Thomas Jefferson to compose the original
draft of the document, which Congress would edit to produce the final version.
The Declaration was ultimately a formal explanation of why Congress had voted on July
2 to declare independence from Great Britain, more than a year after the outbreak of
the American Revolutionary War. The next day, Adams wrote to his wife Abigail: "The
Second Day of July 1776, will be the most memorable Epocha, in the History of America."
But Independence Day is actually celebrated on July 4, the date that the Declaration of
Independence was approved.

After ratifying the text on July 4, Congress issued the Declaration of Independence in
several forms. It was initially published as the printed Dunlap broadside that was widely
distributed and read to the public. The source copy used for this printing has been lost,
and may have been a copy in Thomas Jefferson's hand.[5] Jefferson's original draft, complete
with changes made by John Adams and Benjamin Franklin, and Jefferson's notes of changes made
by Congress, are preserved at the Library of Congress. The best-known version of the Declaration
is a signed copy that is displayed at the National Archives in Washington, D.C., and which is
popularly regarded as the official document. This engrossed copy was ordered by Congress on
July 19 and signed primarily on August 2.

The sources and interpretation of the Declaration have been the subject of much scholarly inquiry.
The Declaration justified the independence of the United States by listing colonial grievances against

```
King George III, and by asserting certain natural and legal rights,
including a right of revolution.
Having served its original purpose in announcing independence,
references to the text of the
Declaration were few in the following years. Abraham Lincoln made it
the centerpiece of his rhetoric
(as in the Gettysburg Address of 1863) and his policies. Since then,
it has become a well-known statement
on human rights, particularly its second sentence:

We hold these truths to be self-evident, that all men are created
equal, that they are endowed by their
Creator with certain unalienable Rights, that among these are Life,
Liberty and the pursuit of Happiness.

This has been called "one of the best-known sentences in the English
language", containing "the most potent
and consequential words in American history". The passage came to
represent a moral standard to which
the United States should strive. This view was notably promoted by
Abraham Lincoln, who considered the
Declaration to be the foundation of his political philosophy and
argued that it is a statement of principles
through which the United States Constitution should be interpreted.

The U.S. Declaration of Independence inspired many other similar
documents in other countries, the first
being the 1789 Declaration of Flanders issued during the Brabant
Revolution in the Austrian Netherlands
(modern-day Belgium). It also served as the primary model for numerous
declarations of independence across
Europe and Latin America, as well as Africa (Liberia) and Oceania (New
Zealand) during the first half of the
19th century.'''

word_list = string_words.split()

word_freq = [word_list.count(n) for n in word_list]

print("String:\n {} \n".format(string_words))
print("List:\n {} \n".format(str(word_list)))
print("Pairs (Words and Frequencies:\n
{}".format(str(list(zip(word_list, word_freq)))))
```


"The quick brown fox jumps over The lazy dog"
Convert the string to a list and print all the words and their frequencies
Converting string to the list
" The quick brown fox jumps over The lazy dog "
Frequency count
1 1 1 1 1 1 2 1 1
[('The', 2), ('quick', 1), ('brown', 1), ('fox', 1), ('jumps', 1), ('over', 1), ('lazy', 1), ('dog', 1)]

7. Write a Python program to count the number of each character of a given text of a text file

Write a Python program to count the number of each character of a text file.

Inputs:
abc.txt -
German Unity Day
From Wikipedia, the free encyclopedia
The Day of German Unity (German: Tag der DeutschenEinheit) is the national day of Germany, celebrated on 3 October as a public holiday. It commemorates the anniversary of German reunification in 1990, when the goal of a united Germany that originated in the middle of the 19th century, was fulfilled again. Therefore, the name addresses neither the re-union nor the union, but the unity of Germany. The Day of German Unity on 3 October has been the German national holiday since 1990, when the reunification was formally completed.

```
File Name:  c:/,….
```

8~~. Write a Python program to get the top stories from Google news~~

```
import bs4
from bs4 import BeautifulSoup as soup
from urllib.request import urlopen

news_url="https://news.google.com/news/rss"
Client=urlopen(news_url)
xml_page=Client.read()
Client.close()
```

```
soup_page=soup(xml_page,"xml")
news_list=soup_page.findAll("item")
# Print news title, url and publish date
for news in news_list:
  print(news.title.text)
  print(news.link.text)
  print(news.pubDate.text)
  print("-"*60)
```

9. Write a Python program to get a list of locally installed Python modules.

```
import pkg_resources
installed_packages = pkg_resources.working_set
installed_packages_list = sorted(["%s==%s" % (i.key, i.version)
     for i in installed_packages])
for m in installed_packages_list:
    print(m)
```

Output

```
-illow==6.1.0
absl-py==0.8.1
apriori==1.0.0
apyori==1.1.1
asgiref==3.2.3
astor==0.8.0
attrs==19.2.0
backcall==0.1.0
bleach==3.1.0
bravado-core==5.13.2
bravado==10.4.2
cachetools==3.1.1
certifi==2019.6.16
chardet==3.0.4
click==7.0
cmake==3.15.3
colorama==0.4.1
cycler==0.10.0
decorator==4.4.0
defusedxml==0.6.0
django==3.0
dnspython==1.16.0
docopt==0.6.2
efficient-apriori==1.0.0
entrypoints==0.3
eventlet==0.25.1
face-recognition-models==0.3.0
future==0.18.1
```

```
gast==0.3.2
gitdb2==2.0.6
gitpython==3.0.4
google-api-python-client==1.7.11
google-auth-httplib2==0.0.3
google-auth==1.6.3
graphviz==0.13
greenlet==0.4.15
grpcio==1.25.0
h5py==2.10.0
hog==0.1.7
httplib2==0.14.0
idna==2.8
imageio==2.6.1
importlib-metadata==0.23
ipykernel==5.1.2
ipython-genutils==0.2.0
ipython==7.8.0
jedi==0.15.1
jinja2==2.10.3
joblib==0.14.0
json5==0.8.5
jsonpointer==2.0
jsonref==0.2
jsonschema==3.1.1
jupyter-client==5.3.4
jupyter-core==4.6.0
jupyterlab-server==1.0.6
jupyterlab==1.1.4
keras-applications==1.0.8
keras-preprocessing==1.1.0
kiwisolver==1.1.0
knn==1.0.0
markdown==3.1.1
markupsafe==1.1.1
matplotlib==3.1.1
metrics==0.3.3
missingno==0.4.2
mistune==0.8.4
mlxtend==0.17.0
monotonic==1.5
more-itertools==7.2.0
mouseinfo==0.0.4
msgpack-python==0.5.6
nbc==1.1.2
nbconvert==5.6.0
nbformat==4.4.0
neptune-client==0.4.92
neptune-notebooks==0.0.12
networkx==2.4
nltk==3.2.4
notebook==6.0.1
numpy==1.17.1
nvidia-ml-py3==7.352.0
```

```
oauth2client==4.1.3
oauthlib==3.1.0
opencv-contrib-python==4.1.1.26
opencv-python==4.1.1.26
pandas==0.25.1
pandocfilters==1.4.2
parso==0.5.1
path.py==12.0.1
pathlib2==2.3.5
pathspec==0.5.5
pickle-mixin==1.0.2
pickleshare==0.7.5
pillow==5.3.0
pip==19.3.1
plot==0.6.4
preprocessing==0.1.13
prometheus-client==0.7.1
prompt-toolkit==2.0.10
protobuf==3.10.0
psutil==5.6.3
py4j==0.10.7
pyalgotrade==0.20
pyasn1-modules==0.2.7
pyasn1==0.4.7
pyautogui==0.9.47
pydotplus==2.0.2
pydrive==1.3.1
pygetwindow==0.0.7
pygments==2.2.0
pyjwt==1.7.1
pymsgbox==1.0.7
pyparsing==2.4.2
pyperclip==1.7.0
pyrect==0.1.4
pyrsistent==0.15.4
pyscreeze==0.1.22
pysocks==1.7.0
pyspark==2.4.4
python-dateutil==2.8.0
pytweening==1.0.3
pytz==2019.2
pywavelets==1.1.1
pywin32==225
pywinpty==0.5.5
pyyaml==5.1.2
pyzmq==18.1.0
requests-oauthlib==1.2.0
requests==2.22.0
resize==0.1.0
retrying==1.3.3
rfc3987==1.3.8
rsa==4.0
scikit-image==0.16.2
scikit-learn==0.21.3
```

```
scipy==1.3.1
seaborn==0.9.0
send2trash==1.5.0
setuptools==41.4.0
shape==1.0.0
simplejson==3.16.0
six==1.12.0
sklearn==0.0
smmap2==2.0.5
sphinx-rtd-theme==0.2.4
sqlparse==0.3.0
strict-rfc3339==0.7
svm==0.1.0
swagger-spec-validator==2.4.3
tensorboard==1.12.2
tensorflow==1.12.0
termcolor==1.1.0
terminado==0.8.2
testpath==0.4.2
textblob==0.15.3
tornado==6.0.3
traitlets==4.3.3
tweepy==3.8.0
typing-extensions==3.7.4
typing==3.7.4.1
uritemplate==3.0.0
urllib3==1.25.3
virtualenv==16.7.7
wcwidth==0.1.7
webcolors==1.10
webencodings==0.5.1
websocket-client==0.56.0
werkzeug==0.16.0
wheel==0.33.6
ws4py==0.5.1
xlrd==1.2.0
xmltodict==0.12.0
yellowbrick==1.0.1
zipp==0.6.0
```

10. Write a Python program to display some information about the OS where the script is running

```
import platform as pl

os_profile = [
        'architecture',
        'linux_distribution',
        'mac_ver',
        'machine',
        'node',
        'platform',
        'processor',
        'python_build',
        'python_compiler',
        'python_version',
        'release',
        'system',
        'uname',
        'version',
    ]
for key in os_profile:
  if hasattr(pl, key):
    print(key +  ": " + str(getattr(pl, key)()))
```

Output

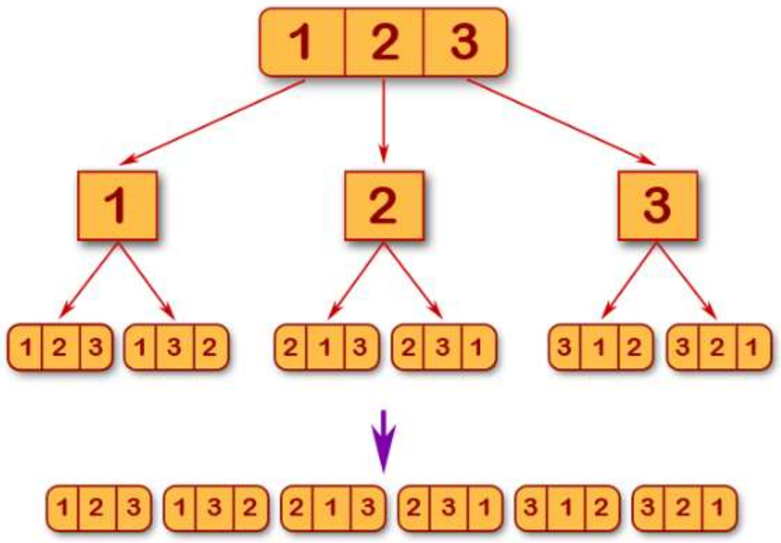
```
architecture: ('32bit', 'WindowsPE')
linux_distribution: ('', '', '')
mac_ver: ('', ('', '', ''), '')
machine: AMD64
node: DESKTOP-UDT9A3S
platform: Windows-10-10.0.18362-SP0
processor: Intel64 Family 6 Model 58 Stepping 9, GenuineIntel
python_build: ('tags/v3.7.4:e09359112e', 'Jul  8 2019 19:29:22')
python_compiler: MSC v.1916 32 bit (Intel)
python_version: 3.7.4
release: 10
system: Windows
uname: uname_result(system='Windows', node='DESKTOP-UDT9A3S', release='10'
, version='10.0.18362', machine='AMD64', processor='Intel64 Family 6 Model
58 Stepping 9, GenuineIntel')
version: 10.0.18362
```

11. Write a Python program to check the sum of three elements (each from an array) from three arrays is equal to a target value. Print all those three-element combinations.

Sample data:
/*
X = [10, 20, 20, 20]
Y = [10, 20, 30, 40]
Z = [10, 30, 40, 20]
target = 70

```
import itertools
from functools import partial
X = [10, 20, 20, 20]
Y = [10, 20, 30, 40]
Z = [10, 30, 40, 20]
T = 70

def check_sum_array(N, *nums):
  if sum(x for x in nums) == N:
    return (True, nums)
  else:
      return (False, nums)
pro = itertools.product(X,Y,Z)
func = partial(check_sum_array, T)
sums = list(itertools.starmap(func, pro))

result = set()
for s in sums:
    if s[0] == True and s[1] not in result:
      result.add(s[1])
      print(result)
```

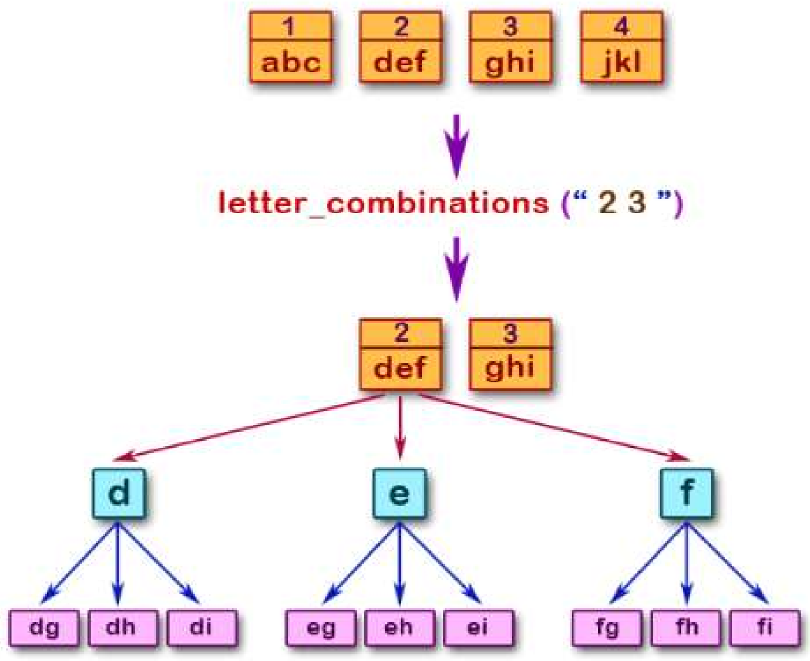
12. Write a Python program to create all possible permutations from a given collection of distinct numbers



```
def permute(nums):
  result_perms = [[]]
  for n in nums:
    new_perms = []
    for perm in result_perms:
      for i in range(len(perm)+1):
        new_perms.append(perm[:i] + [n] + perm[i:])
        result_perms = new_perms
  return result_perms

my_nums = [1,2,3]
print("Original Cofllection: ",my_nums)
print("Collection of distinct numbers:\n",permute(my_nums))
```

13. Write a Python program to get all possible two digit letter combinations from a digit (1 to 9) string.

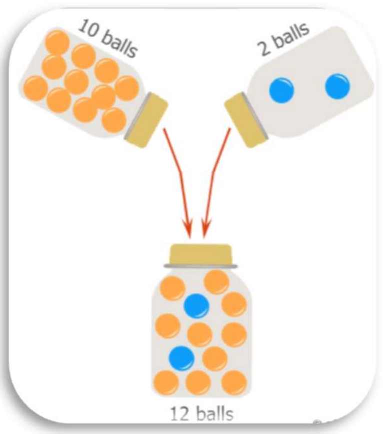
```
string_maps = {
"1": "abc",
"2": "def",
"3": "ghi",
"4": "jkl",
"5": "mno",
"6": "pqrs",
"7": "tuv",
"8": "wxy",
"9": "z"
}
```



```
def letter_combinations(digits):
    if digits == "":
        return []
    string_maps = {
        "1": "abc",
        "2": "def",
        "3": "ghi",
        "4": "jkl",
        "5": "mno",
        "6": "pqrs",
        "7": "tuv",
        "8": "wxy",
        "9": "z"
    }
    result = [""]
    for num in digits:
        temp = []
        for an in result:
            for char in string_maps[num]:
                temp.append(an + char)
        result = temp
    return result

digit_string = "47"
print(letter_combinations(digit_string))
```

```
digit_string = "29"
print(letter_combinations(digit_string))
```

14. Write a Python program to add two positive integers without using **the '+' operator**.



Sum of two bits can be obtained by performing XOR (^) of the two bits. Carry bit can be obtained by performing AND (&) of two bits.

Above is simple Half Adder logic that can be used to add 2 single bits. We can extend this logic for integers. If x and y don't have set bits at same position(s), then bitwise XOR (^) of x and y gives the sum of x and y. To incorporate common set bits also, bitwise AND (&) is used. Bitwise AND of x and y gives all carry bits. We calculate (x & y) << 1 and add it to x ^ y to get the required result.

```
def Add(x, y):

    # Iterate till there is no carry
    while (y != 0):

        # carry now contains common
        # set bits of x and y
        carry = x & y
        # Sum of bits of x and y where at

        # least one of the bits is not set

        x = x ^ y


        # Carry is shifted by one so that

        # adding it to x gives the required sum

        y = carry << 1


    return x


print(Add(15, 32))
```

```
def add_without_plus_operator(a, b):
    while b != 0:
        data = a & b
        a = a ^ b
        b = data << 1
    return a
print(add_without_plus_operator(2, 10))
print(add_without_plus_operator(-20, 10))
print(add_without_plus_operator(-10, -20))
```

Output

```
12

-10

-30
```

15. Write a Python program to check the priority of the four operators (+, -, *, /).

```python
from collections import deque
import re

__operators__ = "+-/*"
__parenthesis__ = "()"
__priority__ = {
    '+': 0,
    '-': 0,
    '*': 1,
    '/': 1,
}

def test_higher_priority(operator1, operator2):
    return __priority__[operator1] >= __priority__[operator2]

print(test_higher_priority('*','-'))
print(test_higher_priority('+','-'))
print(test_higher_priority('+','*'))
print(test_higher_priority('+','/'))
print(test_higher_priority('*','/'))
```

Output

```
True

True

False

False

True
```