In [ ]:
```
# hello world
print("Hello world")
```

```
Hello world
```

In [ ]:
```
# •     To calculate area of a circle, square and triangle
# calculate area of circle
r = int(input("Enter circle's radius length: "))
pi = 3.14
circ_area = pi * r * r
print(f"The area of triangle is {circ_area}.")

# calculate area of rectangle
l = int(input("Enter rectangle's length: "))
b = int(input("Enter rectangle's breadth: "))
rect_area = l * b
print(f"The area of rectangle is {rect_area}.")

# calculate area of square
s = int(input("Enter square's side length: "))
sqt_area = s * s
print(f"The area of square is {sqt_area}.")

# calculate area of triangle
h = int(input("Enter triangle's height length: "))
b = int(input("Enter triangle's breadth length: "))
tri_area = 0.5 * b * h
print(f"The area of triangle is {tri_area}.")
```

```
The area of rectangle is 30.
```

In [ ]:
```
# •     To find out whether a number is Positive, Zero or Negative
num = float(input("Enter a number: "))
if num > 0:
   print("Positive number")
elif num == 0:
   print("Zero")
else:
   print("Negative number")
```

In [ ]:
```
# •     To get today’s date and current time
import datetime

print(datetime.datetime.now())
```

```
2021-10-09 17:46:03.400872
```

In [ ]:
```
# •     To get the version of python on which you are working
import platform
print(platform.python_version())
```

```
3.9.7
```

In [ ]:
```
# •     To convert kilometers to miles
kilometers = float(input("Enter value (km): "))

miles = kilometers * 0.621371
print('%0.2f kilometers is equal to %0.2f miles' %(kilometers,miles))
```

```
25.00 kilometers is equal to 15.53 miles
```

In [ ]:
```
# •     To convert Celsius to Fahrenheit
tempC = float(input("Enter value (celcius): "))
fahrenheit = (tempC * 1.8) + 32
print('%0.1f degree Celsius is equal to %0.1f degree Fahrenheit' %(tempC,fahre
```

```
32.0 degree Celsius is equal to 89.6 degree Fahrenheit
```

In [ ]:
```
# •     To find whether a string is palindrome. Ask user to give the input a
def reverse(s):
    return s[::-1]

UserString = input("Enter the string to check for palindrome")

if(reverse(UserString) == UserString):
    print("Its palindrome")
else:
    print("Not a palindrome")
```

```
Not a palindrome
```

In [ ]:
```
# Write a program to calculate simple interest.
p = float(input("  Principal Amount : "))
r = float(input("  Rate Of Interest : "))
t = float(input(" Time : "))

simple_interest = (p * r * t) / 100

print("\nSimple Interest for Principal Amount %0.2f = %0.2f" %(p, simple_inte
```

```
Simple Interest for Principal Amount 5.00 = 1.25
```

In [ ]:
```
# •     Write a Python program that accepts an integer (n) and computes the va
i=int(input("Enter a number:"))
num= (i+ ((i*10)+i) + ((i*100)+(i*10)+i))
print(num)

#or
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)
```

```
615
615
```

In [ ]:

```
# •     Write a Python program to sum of three given integers. However, if tw
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(1, 2, 2))
print(sum(4, 2, 3))
```

```
0
9
```

In [ ]:

```
# •     Write a Python program to convert the distance (in feet) to inches, y
ft = int(input("Input distance in feet: "))
inches = ft * 12
yards = ft / 3.0
miles = ft / 5280.0

print(" %i inches." % inches)
print(" %.2f yards." % yards)
print(" %.2f miles." % miles)
```

```
 60 inches.
 1.67 yards.
 0.00 miles.
```

In [ ]:

In [ ]:

```
# )Write a Python program to construct the following pattern, using a nested 
# *

# * *

# * * *

# * * * *

# * * * * *

# * * * *

# * * *

# * *

# *

n=5
for i in range(n):
    for j in range(i):
        print ('* ', end="")
    print('')

for i in range(n,0,-1):
    for j in range(i):
        print('* ', end="")
    print('')
```

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

In [ ]:

```
#  Find numbers which are divisible by 7 and multiple of 5 between a range Fi
nl=[]
for x in range(1500, 2701):
    if (x%7==0) and (x%5==0):
        print(x)
```

```
1505
1540
1575
1610
1645
1680
1715
1750
1785
1820
1855
```

```
1890
1925
1960
1995
2030
2065
2100
2135
2170
2205
2240
2275
2310
2345
2380
2415
2450
2485
2520
2555
2590
2625
2660
2695
```

In [ ]:

```
# Write a Python program to count the number of even and odd numbers from a se
# numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)
# Expected Output :
# Number of even numbers : 4
# Number of odd numbers : 5
numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)
odd = 0
even = 0
for x in numbers:
        if not x % 2:
             even+=1
        else:
             odd+=1
print("Number of even numbers :",even)
print("Number of odd numbers :",odd)
```

```
Number of even numbers : 4
Number of odd numbers : 5
```

In [ ]:

```
# Write a Python program that prints all the numbers from 0 to 6 except 3 and
# Note : Use 'continue' statement.
for x in range(6):
    if (x == 3 or x==6):
        continue
    print(x,end=' ')
print("\n")
```

```
0 1 2 4 5
```

In [ ]:

```
# Write a Python program to get the Fibonacci series between 0 to 50.
x,y=0,1

while y<50:
    print(y)
    x,y = y,x+y
```

```
1
1
2
3
5
8
13
21
34
```

In [ ]:

```
# Write a Python program to print alphabet pattern 'A'.
result_str="";
for row in range(0,7):
    for column in range(0,7):
        if (((column == 1 or column == 5) and row != 0) or ((row == 0 or row =
            result_str=result_str+"*"
        else:
            result_str=result_str+" "
    result_str=result_str+"\n"
print(result_str)
```

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

In [ ]:

```
# Write a program to check whether a number is Prime number or not.
num = int(input("Enter a number : "))
if num > 1:

    for i in range(2, int(num/2)+1):
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")

else:
    print(num, "is not a prime number")
```

```
25 is not a prime number
```

```
In [ ]:
```

```python
# factorial
def factorial(n):
    return 1 if (n==1 or n==0) else n * factorial(n - 1)


num = int(input("Enter a number: "))
print ("Factorial of",num,"is",factorial(num))
```

```
Factorial of 20 is 2432902008176640000
```

```
In [ ]:
```

```python
# Multiplication table (from 1 to 10) in Python

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

for i in range(1, 11):
   print(num, 'x', i, '=', num*i)
```

```
25 x 1 = 25
25 x 2 = 50
25 x 3 = 75
25 x 4 = 100
25 x 5 = 125
25 x 6 = 150
25 x 7 = 175
25 x 8 = 200
25 x 9 = 225
25 x 10 = 250
```

```
In [ ]:
```

```python
#Armstrong Number
lower = int(input("Enter the lower limit"))
upper = int(input("Enter the upper limit"))

for num in range(lower, upper + 1):
#    print(str(num))
   order = len(str(num))

   sum = 0

   temp = num
   while temp > 0:
       digit = temp % 10
       sum += digit ** order
    #    print(temp / 10)
       temp = temp // 10

   if num == sum:
       print(num)
```

In [ ]:

```
# Python Program to find the factors of a number

def factors(n):
   print("The factors of",n,"are:")
   for i in range(1, n + 1):
       if n % i == 0:
           print(i)

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

factors(num)
```

```
The factors of 25 are:
1
5
25
```

In [ ]:

```
# HCF and LCM
def get_gcd(a,b):
    gcd = 1
    for i in range(1,a+1):
        if a%i==0 and b%i==0:
            gcd = i
    return gcd


first = int(input('Enter first number: '))
second = int(input('Enter second number: '))
gcd = get_gcd(first, second)
print('HCF or GCD of %d and %d is %d' %(first, second,gcd ))


lcm = first * second / gcd
print('LCM of %d and %d is %d' %(first, second, lcm))
```

```
HCF or GCD of 25 and 63 is 1
LCM of 25 and 63 is 1575
```

In [ ]:

```
# •     Write a Python program to sum all the items in a list
def sum_list(items):
    sum_numbers = 0
    for x in items:
        sum_numbers += x
    return sum_numbers
print(sum_list([5,2,-3]))
```

```
4
```

```
In [ ]:
```

```
# •     Write a Python program to get the largest and smallest number from a 
numList = [4,56,65,654,654,464,22,3]
print("\nThe Smallest Element in this List is: ", min(numList))
print("The Largest Element in this List is: ", max(numList))

# or
numList.sort()

print("\nThe Smallest Element in this List is : ", numList[0])
print("The Largest Element in this List is : ", numList[- 1])
```

```
The Smallest Element in this List is:  3
The Largest Element in this List is:  654

The Smallest Element in this List is :  3
The Largest Element in this List is :  654
```

```
In [ ]:
```

```
# •     Write a Python program to check a list is empty or not.
l = []
if not l:
  print("List is empty")

# or



if len(l) == 0:
    print("List is empty")
else:
    print("List is not empty")
```

```
List is empty
List is empty
```

```
In [ ]:
```

```
# •     Write a Python program to clone or copy a list.
listo = [11, 23, 45, 24, 5]
listn = list(listo)
print(listo)
print(listn)
```

```
[11, 23, 45, 24, 5]
[11, 23, 45, 24, 5]
```

```
In [ ]:
```

```
# •     Write a Python program to print a specified list after removing the 0
color = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
color = [x for (i,x) in enumerate(color) if i not in (0,4,5)]
print(color)
```

```
['Green', 'White', 'Black']
```

In [ ]:

```
# •     Write a Python program access the index of a list.
l = [3, 45, 85, 81, 113]
for num, val in enumerate(l):
    print(num, val)
```

```
0 3
1 45
2 85
3 81
4 113
```

In [ ]:

```
# •     Write a Python program to append a list to the second list.
list1 = [1, 2, 3, 0]
list2 = ['Red', 'Green', 'Black']
final_list = list1 + list2
print(final_list)
```

```
[1, 2, 3, 0, 'Red', 'Green', 'Black']
```

In [ ]:

```
# •     Write a Python program to select an item randomly from a list.
import random
color_list = ['Red', 'Blue', 'Green', 'White', 'Black']
print(random.choice(color_list))
```

```
White
```

In [ ]:

```
# •     Write a Python program to get unique values from a list.
nList = [10, 20, 30, 40, 20, 50, 60, 40]
uList = []

for x in nList:
    if x not in uList:
            uList.append(x)
print(uList)

# or

print("Original List : ",nList)
my_set = set(nList)
print(my_set)
my_new_list = list(my_set)
print("List of unique numbers : ",my_new_list)
```

```
[10, 20, 30, 40, 50, 60]
Original List :  [10, 20, 30, 40, 20, 50, 60, 40]
{40, 10, 50, 20, 60, 30}
List of unique numbers :  [40, 10, 50, 20, 60, 30]
```

In [ ]:

```
# •     Write a Python program to get the second largest and second smallest
numList = [4,56,65,654,654,464,22,3]
numList.sort()
print("Second Smallest : %i"  %numList[1])
print("Second Largest : %i" %numList[-2])
```

```
Second Smallest : 4
Second Largest : 654
```

In [ ]:

```
# •     Write a Python program to remove duplicates from a list.
nList = [10, 20, 30, 40, 20, 50, 60, 40]
print("Original List : ",nList)
my_set = set(nList)
# print(my_set)
nList = list(my_set)
print("New list : ",nList)
```

```
Original List :  [10, 20, 30, 40, 20, 50, 60, 40]
New list :  [40, 10, 50, 20, 60, 30]
```

In [ ]:

```
# count characters
test_str = input("Enter a string : ")

# using naive method to get count
# of each element in string
all_freq = {}

for i in test_str:
    if i in all_freq:
        all_freq[i] += 1
    else:
        all_freq[i] = 1

# printing result
print ("Count of all characters in given string is :\n "
                                               +  str(all_freq))
```

```
Count of all characters in given string is :
 {'d': 1, 'e': 2, 'p': 1, 'a': 1, 'k': 1}
```

In [ ]:

```
# •     Write a Python program to get a single string from two given strings,
def swapFirstTwoCharacters(a, b):
  str1 = b[:2] + a[2:]
  str2 = a[:2] + b[2:]

  return str1 + ' ' + str2

str1 = input("Enter string 1 : ")

str2 = input("Enter string 2 : ")
print(swapFirstTwoCharacters(str1, str2))
```

```
paepak dewade
```

In [ ]:
```
# •     Write a Python program to add 'ing' at the end of a given string (len
def add_suffix(str1):
  length = len(str1)

  if length > 2:
    if str1[-3:] == 'ing':
      str1 += 'ly'
    else:
      str1 += 'ing'

  return str1

Str = input("Enter a string : ")
print(add_suffix(Str))
```

```
helloingly
```

In [ ]:
```
# •     Write a Python program to find the first appearance of the substring
def not_poor(str1):
  snot = str1.find('not')
  spoor = str1.find('poor')


  if spoor > snot and snot>0 and spoor>0:
    str1 = str1.replace(str1[snot:(spoor+4)], 'good')
    return str1
  return str1

Str = input("Enter a String : ")
print(not_poor(Str))
```

```
its good
```

In [ ]:
```

```

In [ ]:
```
# tuple
my_tuple = ("red", "blue", "green")
print(my_tuple)
```

```
('red', 'blue', 'green')
```

In [ ]:
```
#Create a tuple with different data types
tuple1 = ("tuple", False, 3.2, 1)
print(tuple1)
```

```
('tuple', False, 3.2, 1)
```

In [ ]:
```
#Create a tuple with numbers
tuplex = 5, 10, 15, 20, 25
print(tuplex[0])
```

```
5
```

In [ ]:

```
# add item in a tuple
tuple1 = (4, 6, 2, 8, 3, 1)
print(tuple1)

tuple1 = tuple1 + (66,)
print(tuple1)

tuplex = tuplex[:5] + (15, 20, 25) + tuplex[:5]
print(tuplex)

listx = list(tuplex)

listx.append(30)
tuplex = tuple(listx)
print(tuplex)
```

```
(4, 6, 2, 8, 3, 1)
(4, 6, 2, 8, 3, 1, 66)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3, 30)
```

In [ ]:

```
# •     Write a Python program to get the 4th element and 4th element from la
tuple1 = (4, 6, 2, 8, 3, 1)
print(tuple1[3])
print(tuple1[-4])
```

```
8
2
```

In [ ]:

```
# •     Write a Python program to check whether an element exists within a tu
tuple1 = (4, 6, 2, 8, 3, 1)
print(5 in tuple1)
print(4 in tuple1)
```

```
False
True
```

In [ ]:

```
# •     Write a Python program to convert a list to a tuple.
tuple1 = (4, 6, 2, 8, 3, 1)
print(list(tuple1))
```

```
[4, 6, 2, 8, 3, 1]
```

In [ ]:

```
# •     Write a Python program to slice a tuple.

tuple1 = (4, 6, 2, 8, 3, 1)
tuple2 = tuple1[3:5]
print(tuple2)
tuple3 = tuple1[:4]
print(tuple3)
```

```
(8, 3)
(4, 6, 2, 8)
```

In [ ]:

```
# •     Write a Python program to find the index of an item of a tuple.
tuple1 = (4, 6, 2, 8, 3, 1)
print(tuple1.index(3))
```

```
4
```

In [ ]:

```
# •     Write a Python program to find the length of a tuple.
tuple1 = (4, 6, 2, 8, 3, 1)
print(len(tuple1))
```

```
6
```

In [ ]:

```
# •     Write a Python program to sort a tuple by its float element
price = [('item1', '12.20'), ('item2', '15.10'), ('item3', '24.5')]
print( sorted(price, key=lambda x: float(x[1]), reverse=True))
```

```
[('item3', '24.5'), ('item2', '15.10'), ('item1', '12.20')]
```

In [ ]:

```
# •     Write a Python script to sort (ascending and descending) a dictionary
import operator
d = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
print('Original dictionary : ',d)
sorted_d = sorted(d.items(), key=operator.itemgetter(1))
print('Dictionary in ascending order by value : ',sorted_d)
sorted_d = dict( sorted(d.items(), key=operator.itemgetter(1),reverse=True))
print('Dictionary in descending order by value : ',sorted_d)
```

```
Original dictionary :  {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
Dictionary in ascending order by value :  [(0, 0), (2, 1), (1, 2), (4, 3), (3,
4)]
Dictionary in descending order by value :  {3: 4, 4: 3, 1: 2, 2: 1, 0: 0}
```

In [ ]:

```
# •     Write a Python script to add a key to a dictionary
d = {0:10, 1:20}
print(d)
d.update({2:30})
print(d)
```

```
{0: 10, 1: 20}
{0: 10, 1: 20, 2: 30}
```

In [ ]:

```
# concatenate dictionaries
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

```
In [ ]:
# check if key present
def checkKey(dict, key):

    if key in dict.keys():
        print("\"" + key + "\"" +" Present, ", end =" ")
        print("value =", dict[key])
    else:
        print("\"" +key + "\"" +" Not present")

dict = {'a': 100, 'b':200, 'c':300}

key = 'b'
checkKey(dict, key)

key = 'w'
checkKey(dict, key)
```

```
"b" Present,  value = 200
"w" Not present
```

```
In [ ]:
# •     Write a Python program to iterate over dictionaries using for loops.

d = {'Red': 1, 'Green': 2, 'Blue': 3}
for color_key, value in d.items():
     print(color_key, " : ", d[color_key])
```

```
Red  :  1
Green  :  2
Blue  :  3
```

```
In [ ]:
# •     Write a Python script to generate and print a dictionary that contain
n=int(input("Input a number "))
d = {}

for x in range(1,n+1):
    d[x]=x*x

print(d)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
```

```
In [ ]:
# •     Write a Python script to print a dictionary where the keys are number
d={}
for x in range(1,16):
    d[x]=x**2
print(d)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121,
12: 144, 13: 169, 14: 196, 15: 225}
```

In [ ]:

```
# •     Write a Python script to merge two Python dictionaries.
d1 = {'a': 100, 'b': 200}
d2 = {'x': 300, 'y': 200}
d = d1.copy()
d.update(d2)
print(d)
```

```
{'a': 100, 'b': 200, 'x': 300, 'y': 200}
```

In [ ]:

```
# Write a Python program to sum all the items in a dictionary.
my_dict = {'data1':100,'data2':554,'data3':-247}
print(sum(my_dict.values()))

# or

def returnSum(myDict):

    list = []
    for i in myDict:
        list.append(myDict[i])
    final = sum(list)

    return final

dict = {'a': 100, 'b':200, 'c':300}
print("Sum :", returnSum(dict))
```

```
407
Sum : 600
```

In [ ]:

```
# remove a key
myDict = {'a':1,'b':2,'c':3,'d':4}
print(myDict)
if 'a' in myDict:
    del myDict['a']
print(myDict)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
{'b': 2, 'c': 3, 'd': 4}
```

In [ ]:

```
# •     Write a Python program to get the maximum and minimum value in a dict.
my_dict = {'x':500, 'y':5874, 'z': 560}

key_max = max(my_dict.keys(), key=(lambda k: my_dict[k]))
key_min = min(my_dict.keys(), key=(lambda k: my_dict[k]))

print('Maximum Value: ',my_dict[key_max])
print('Minimum Value: ',my_dict[key_min])
```

```
Maximum Value:  5874
Minimum Value:  500
```

```
In [ ]: # •     Write a Python program to combine two dictionary adding values for co
        dict1 = {'a': 12, 'for': 25, 'c': 9}
        dict2 = {'a': 100, 'd': 200, 'c': 300}

        for key in dict2:
            if key in dict1:
                dict2[key] = dict2[key] + dict1[key]
            else:
                pass

        print(dict2)
```

```
{'a': 112, 'd': 200, 'c': 309}
```

```
In [ ]: # •     Write a Python program to print all unique values in a dictionary.
        L = [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S00
        print("Original List: ",L)
        uniqueValues = set( val for dic in L for val in dic.values())
        print("Unique Values: ",uniqueValues)
```

```
Original List:  [{'V': 'S001'}, {'V': 'S002'}, {'VI': 'S001'}, {'VI': 'S005'},
{'VII': 'S005'}, {'V': 'S009'}, {'VIII': 'S007'}]
Unique Values:  {'S007', 'S009', 'S001', 'S005', 'S002'}
```

```
In [ ]: # •     Write a Python program to get the top three items in a shop.
        from heapq import nlargest
        from operator import itemgetter
        items = {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}
        for name, value in nlargest(3, items.items(), key=itemgetter(1)):
            print(name, value)
```

```
item4 55
item1 45.5
item3 41.3
```

```
In [ ]: # •     Write a Python program to create a set.
        my_set = {1, 2, 3}
        print(my_set)

        # set of mixed datatypes
        my_set = {1.0, "Hello", (1, 2, 3)}
        print(my_set)
```

```
{1, 2, 3}
{1.0, (1, 2, 3), 'Hello'}
```

```
In [ ]: # •     Write a Python program to iterate over sets.
        my_set = {1.0, "Hello", (1, 2, 3)}

        # Iterating using for loop
        for val in my_set:
            print(val)
```

```
1.0
(1, 2, 3)
```

```
Hello
```

In [ ]:

```
# •     Write a Python program to add member(s) in a set.
my_set = {1.0, "Hello", (1, 2, 3)}
my_set.add(11)
print(my_set)
my_set.add("ABC")
print(my_set)
```

```
{11, 1.0, (1, 2, 3), 'Hello'}
{'ABC', 1.0, 11, (1, 2, 3), 'Hello'}
```

In [ ]:

```
# •     Write a Python program to remove item(s) from set
my_set = {1.0, "Hello", (1, 2, 3)}
my_set.add(11)
print(my_set)
my_set.add("ABC")
print(my_set)
my_set.remove("ABC")
print(my_set)
```

```
{11, 1.0, (1, 2, 3), 'Hello'}
{'ABC', 1.0, 11, (1, 2, 3), 'Hello'}
{1.0, 11, (1, 2, 3), 'Hello'}
```

In [ ]:

```
# •     Write a Python program to remove an item from a set if it is present
my_set = {1.0, "Hello", (1, 2, 3)}
my_set.add(11)
print(my_set)
my_set.add("ABC")
print(my_set)

if "ABC" in my_set:
    my_set.remove("ABC")

print(my_set)
```

```
{11, 1.0, (1, 2, 3), 'Hello'}
{'ABC', 1.0, 11, (1, 2, 3), 'Hello'}
{1.0, 11, (1, 2, 3), 'Hello'}
```

In [ ]:
```
# •     Write a Python program to create an intersection of sets.
setx = {1, 2, 3, "Hello"}
sety = {1.0, "Hello", (1,2,3)}
print("Original set elements:")
print(setx)
print(sety)
print("\nIntersection of two said sets:")
result = setx.intersection(sety)
print(result)
```

```
Original set elements:
{1, 2, 3, 'Hello'}
{1.0, (1, 2, 3), 'Hello'}

Intersection of two said sets:
{1.0, 'Hello'}
```

In [ ]:
```
# •     Write a Python program to create a union of sets.
setx = {1, 2, 3, "Hello"}
sety = {1.0, "Hello", (1,2,3)}
print("Original set elements:")
print(setx)
print(sety)
print("\nIntersection of two said sets:")
result = setx.union(sety)
print(result)
```

```
Original set elements:
{1, 2, 3, 'Hello'}
{1.0, (1, 2, 3), 'Hello'}

Intersection of two said sets:
{1, 2, 3, (1, 2, 3), 'Hello'}
```

In [ ]:
```
# •     Write a Python program to create set difference.
setx = {1, 2, 3, "Hello"}
sety = {1.0, "Hello", (1,2,3)}
print("Original set elements:")
print(setx)
print(sety)
print("\ndifference of two said sets: first x-y then y-x")
result = setx.difference(sety)
print(result)

result = sety.difference(setx)
print(result)
```

```
Original set elements:
{1, 2, 3, 'Hello'}
{1.0, (1, 2, 3), 'Hello'}

difference of two said sets: first x-y then y-x
{2, 3}
{(1, 2, 3)}
```

In [ ]:

```
# •     Write a Python program to create a symmetric difference.\
setx = {1, 2, 3, "Hello"}
sety = {1.0, "Hello", (1,2,3)}
print("Original set elements:")
print(setx)
print(sety)
print("\ndifference of two said sets: first x-y then y-x")
result = setx.symmetric_difference(sety)
print(result)

result = sety.symmetric_difference(setx)
print(result)
```

```
Original set elements:
{1, 2, 3, 'Hello'}
{1.0, (1, 2, 3), 'Hello'}

difference of two said sets: first x-y then y-x
{2, 3, (1, 2, 3)}
{2, 3, (1, 2, 3)}
```

In [ ]:

```
# •     Write a Python program to issubset and issuperset.
A = {1, 2, 3, 4, 5}
B = {1, 2, 3}
C = {1, 2, 3}

print(A.issuperset(B))
print(B.issuperset(A))
print(B.issubset(A))
print(C.issuperset(B))
```

```
True
False
True
True
```

In [ ]:

```
# •     Write a Python program to create a shallow copy of sets.
my_set = {1, 2, 3}
my_set2 = {1.0, "Hello", (1, 2, 3)}
set3 = my_set.copy()
print(set3)
```

```
{1, 2, 3}
```

In [ ]:

```
# •     Write a Python program to clear a set.
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
my_set.clear()
print(my_set)
```

```
{1.0, (1, 2, 3), 'Hello'}
set()
```

In [ ]:
```
#write a Python function to find the Max of three numbers.
def max_of_two( x, y ):
    if x > y:
        return x
    return y
def max_of_three( x, y, z ):
    return max_of_two( x, max_of_two( y, z ) )
print(max_of_three(5, 7, -9))
```

```
7
```

In [ ]:
```
#Write a Python function to sum all the numbers in a list. Sample List : (8, 
#Expected Output : 20
def sum(numbers):
    total = 0
    for x in numbers:
        total += x
    return total
print(sum((8, 2, 3, 0, 7)))
```

```
20
```

In [ ]:
```
#Write a Python function to multiply all the numbers in a list. Sample List :
#Expected Output : -336
def multiply(numbers):
    total = 1
    for x in numbers:
        total *= x
    return total
print(multiply((8, 2, 3, -1, 7)))
```

```
-336
```

In [ ]:
```
#Write a Python function that accepts a string and calculate the number of up
#Sample String : 'The quick Brow Fox'
#Expected Output :
#No. of Upper case characters : 3
#No. of Lower case Characters : 12
def string_test(s):
    d={"UPPER_CASE":0, "LOWER_CASE":0}
    for c in s:
        if c.isupper():
           d["UPPER_CASE"]+=1
        elif c.islower():
           d["LOWER_CASE"]+=1
        else:
           pass
    print ("Original String : ", s)
    print ("No. of Upper case characters : ", d["UPPER_CASE"])
    print ("No. of Lower case Characters : ", d["LOWER_CASE"])

string_test('The quick Brown Fox')
```

```
Original String :  The quick Brown Fox
No. of Upper case characters :  3
```

In [ ]:
```
#Write a Python function that checks whether a passed string is palindrome or
def isPalindrome(UserString):
    def reverse(s):
        return s[::-1]

    if(reverse(UserString) == UserString):
        print("Its palindrome")
    else:
        print("Not a palindrome")

UserString = input("Enter the string to check for palindrome")
isPalindrome(UserString)
```

```
Its palindrome
```

In [ ]:
```
#Write a Python function that prints out the first n rows of Pascal's triangle
def pascal_triangle(n):
   trow = [1]
   y = [0]
   for x in range(max(n,0)):
      print(trow)
      trow=[l+r for l,r in zip(trow+y, y+trow)]
   return n>=1
pascal_triangle(6)
```

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
```

Out[ ]:
```
True
```

In [ ]:
```
#Write a Python program that accepts a hyphen-separated sequence of words as
#Sample Items : green-red-yellow-black-white.
#Expected Result : black-green-red-white-yellow
items=[n for n in input().split('-')]
items.sort()
print('-'.join(items))
```

```
black-green-red-white-yellow
```

In [ ]:
```
#Write a Python function to create and print a list where the values are squa
def printValues():
        l = list()
        for i in range(1,21):
                l.append(i**2)
        print(l)

printValues()
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324,
361, 400]
```

In [ ]:
```
#Write a Python program to make a chain of function decorators (bold, italic,
def make_bold(fn):
    def wrapped():
        return "<b>" + fn() + "</b>"
    return wrapped

def make_italic(fn):
    def wrapped():
        return "<i>" + fn() + "</i>"
    return wrapped

def make_underline(fn):
    def wrapped():
        return "<u>" + fn() + "</u>"
    return wrapped
@make_bold
@make_italic
@make_underline
def hello():
    return "hello world"
print(hello())
```

```
<b><i><u>hello world</u></i></b>
```

In [ ]:
```
# Write a NumPy program to test whether any of the elements of a given array
import numpy as np
x = np.array([1, 0, 0, 0])
print(np.any(x))
x = np.array([0, 0, 0, 0])
print(np.any(x))
```

```
True
False
```

In [ ]:
```
#Write a NumPy program to calculate the difference between the maximum and the
#Expected Output:Original array:[[ 0 1 2 3 4 5] [ 6 7 8 9 10 11]]
#Difference between the maximum and the minimum values of the said array:[5 5
import numpy as np
x = np.arange(12).reshape((2, 6))
print("\nOriginal array:")
print(x)
r1 = np.ptp(x, 1)
r2 = np.amax(x, 1) - np.amin(x, 1)
assert np.allclose(r1, r2)
print("\nDifference between the maximum and the minimum values of the array:"
print(r1)
```

```
Original array:
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]

Difference between the maximum and the minimum values of the array:
[5 5]
```

In [ ]:

```
#Write a NumPy program to compute the 80th percentile for all elements in a g
#Expected Output:
#Original array:[1.0, 2.0, 3.0, 4.0]
#Largest integer smaller or equal to the division of the inputs:
# [ 0. 1. 2. 2.]
import numpy as np
x = [1., 2., 3., 4.]
print("Original array:")
print(x)
print("Largest integer smaller or equal to the division of the inputs:")
print(np.floor_divide(x, 1.5))
```

```
Original array:
[1.0, 2.0, 3.0, 4.0]
Largest integer smaller or equal to the division of the inputs:
[0. 1. 2. 2.]
```

In [ ]:

```
#Write a NumPy program to compute the median of flattened given array.
# Note: First array elements raised to powers from second array
#Expected Output:Original array:[[ 0 1 2 3 4 5][ 6 7 8 9 10 11]]
#Median of said array:5.5
import numpy as np
x = np.arange(12).reshape((2, 6))
print("\nOriginal array:")
print(x)
r1 =  np.median(x)
print("\nMedian of said array:")
print(r1)
```

```
Original array:
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]

Median of said array:
5.5
```

In [ ]:

```
#Write a NumPy program to compute the mean, standard deviation, and variance
#Sample output:Original array:[0 1 2 3 4 5]
#Mean: 2.5
# std: 1
# variance: 2.9166666666666665
import numpy as np
x = np.arange(6)
print("\nOriginal array:")
print(x)
r1 = np.mean(x)
r2 = np.average(x)
assert np.allclose(r1, r2)
print("\nMean: ", r1)
r1 = np.std(x)
r2 = np.sqrt(np.mean((x - np.mean(x)) ** 2 ))
assert np.allclose(r1, r2)
print("\nstd: ", 1)
r1= np.var(x)
r2 = np.mean((x - np.mean(x)) ** 2 )
assert np.allclose(r1, r2)
print("\nvariance: ", r1)
```

```
Original array:
[0 1 2 3 4 5]

Mean:  2.5

std:  1

variance:  2.9166666666666665
```

In [ ]:

```
#Write a NumPy program to compute the weighted average of a given array.
#Sample Output:Original array:[0 1 2 3 4]
#Weighted average of the said array:2.6666666666666665
import numpy as np
x = np.arange(5)
print("\nOriginal array:")
print(x)
weights = np.arange(1, 6)
r1 = np.average(x, weights=weights)
r2 = (x*(weights/weights.sum())).sum()
assert np.allclose(r1, r2)
print("\nWeighted average of the said array:")
print(r1)
```

```
Original array:
[0 1 2 3 4]

Weighted average of the said array:
2.6666666666666665
```

In [ ]:

```
#Write a NumPy program to compute the covariance matrix of two given arrays.
#Sample Output:Original array1:[0 1 2] Original array1:[2 1 0]
#Covariance matrix of the said arrays:[[ 1. -1.] [-1. 1.]]
import numpy as np
x = np.array([0, 1, 2])
y = np.array([2, 1, 0])
print("\nOriginal array1:")
print(x)
print("\nOriginal array1:")
print(y)
print("\nCovariance matrix of the said arrays:\n",np.cov(x, y))
```

```
Original array1:
[0 1 2]

Original array1:
[2 1 0]

Covariance matrix of the said arrays:
 [[ 1. -1.]
 [-1.  1.]]
```

In [ ]:

```
# Write a NumPy program to compute cross-correlation of two given arrays.
#Sample Output:Original array1:[0 1 3] Original array1:[2 4 5]
#Cross-correlation of the said arrays:[[2.33333333 2.16666667] [2.16666667 2..
import numpy as np
x = np.array([0, 1, 3])
y = np.array([2, 4, 5])
print("\nOriginal array1:")
print(x)
print("\nOriginal array1:")
print(y)
print("\nCross-correlation of the said arrays:\n",np.cov(x, y))
```

```
Original array1:
[0 1 3]

Original array1:
[2 4 5]

Cross-correlation of the said arrays:
 [[2.33333333 2.16666667]
 [2.16666667 2.33333333]]
```

In [ ]:

```
#Write a NumPy program to compute pearson product-moment correlation coefficie
#Sample Output:Original array1:[0 1 3] Original array1:[2 4 5]
#Pearson product-moment correlation coefficients of the said arrays:[[1. 0.92
# [0.92857143 1. ]]
import numpy as np
x = np.array([0, 1, 3])
y = np.array([2, 4, 5])
print("\nOriginal array1:")
print(x)
print("\nOriginal array1:")
print(y)
print("\nPearson product-moment correlation coefficients of the said arrays:\
```

```
Original array1:
[0 1 3]

Original array1:
[2 4 5]

Pearson product-moment correlation coefficients of the said arrays:
 [[1.         0.92857143]
 [0.92857143 1.        ]]
```

In [ ]:

```
# Write a NumPy program to create an element-wise comparison (greater, greate
import numpy as np
x = np.array([3, 5])
y = np.array([2, 5])
print("Original numbers:")
print(x)
print(y)
print("Comparison - greater")
print(np.greater(x, y))
print("Comparison - greater_equal")
print(np.greater_equal(x, y))
print("Comparison - less")
print(np.less(x, y))
print("Comparison - less_equal")
print(np.less_equal(x, y))
```

```
Original numbers:
[3 5]
[2 5]
Comparison - greater
[ True False]
Comparison - greater_equal
[ True  True]
Comparison - less
[False False]
Comparison - less_equal
[False  True]
```

In [ ]:

```
# Write a NumPy program to test whether two arrays are element-wise equal wit
import numpy as np
print("Test if two arrays are element-wise equal within a tolerance:")
print(np.allclose([1e10,1e-7], [1.00001e10,1e-8]))
print(np.allclose([1e10,1e-8], [1.00001e10,1e-9]))
print(np.allclose([1e10,1e-8], [1.0001e10,1e-9]))
print(np.allclose([1.0, np.nan], [1.0, np.nan]))
print(np.allclose([1.0, np.nan], [1.0, np.nan], equal_nan=True))
```

```
Test if two arrays are element-wise equal within a tolerance:
False
True
False
False
True
```

In [ ]:

```
# Write a NumPy program to test whether none of the elements of a given array
import numpy as np
x = np.array([1, 2, 3, 4])
print(np.all(x))


x = np.array([0, 1, 2, 3])
print(np.all(x))
```

```
True
False
```

In [ ]:

```
# Write a NumPy program to create an array with the values 1, 7, 13, 105 and
import numpy as np
X = np.array([1, 7, 13, 105])
print("Original array:")
print(X)
print("Size of the memory occupied by the said array:")
print("%d bytes" % (X.size * X.itemsize))
```

```
Original array:
[  1   7  13 105]
Size of the memory occupied by the said array:
16 bytes
```

In [ ]:

```
# Write a NumPy program to create an array of 10 zeros,10 ones, 10 fives.
import numpy as np
array=np.zeros(10)
print("An array of 10 zeros:")
print(array)
array=np.ones(10)
print("An array of 10 ones:")
print(array)
array=np.ones(10)*5
print("An array of 10 fives:")
print(array)
```

```
An array of 10 zeros:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
An array of 10 ones:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
An array of 10 fives:
```

In [ ]:

```
# Write a NumPy program to test element-wise for positive or negative infinit
import numpy as np, math
a = np.array([1, 0,math.inf, -math.inf, float('inf'), float('-inf')])
print(np.isinf(a))
```

```
[False False  True  True  True  True]
```

In [ ]:

```
#1.     Write a Pandas program to add, subtract, multiple and divide two Pand
#Sample Series: [2, 4, 6, 8, 10], [1, 3, 5, 7, 9]
import pandas as pd
d1 = pd.Series([2, 4, 6, 8, 10])
d2 = pd.Series([1, 3, 5, 7, 9])
ds = d1 + d2
print("Add two Panda Series:")
print(ds)
print("Subtract two Panda Series:")
ds = d1 - d2
print(ds)
print("Multiply two Panda Series:")
ds = d1 * d2
print(ds)
print("Divide two Panda Series:")
ds = d1 / d2
print(ds)
```

```
Add two Panda Series:
0     3
1     7
2    11
3    15
4    19
dtype: int64
Subtract two Panda Series:
0    1
1    1
2    1
3    1
4    1
dtype: int64
Multiply two Panda Series:
0     2
1    12
2    30
3    56
4    90
dtype: int64
Divide two Panda Series:
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```

In [ ]:

```
#2.     Write a Pandas program to compare the elements of the two Pandas Serie
#Sample Series: [2, 4, 6, 8, 10], [1, 3, 5, 7, 10]
import pandas as pd
s1 = pd.Series([2, 4, 6, 8, 10])
s2 = pd.Series([1, 3, 5, 7, 10])
print("Compare the elements of two Series:")
print("Equals:")
print(s1 == s2)
```

```
Compare the elements of two Series:
Equals:
0    False
1    False
2    False
3    False
4     True
dtype: bool
```

In [ ]:

```
#Write a Pandas program to change the data type of given a column or a Series
import pandas as pd
s1 = pd.Series(['11', '22', 'swift', '30.82', '33'])
print(s1)
print("Change the data type to numeric:")
s2 = pd.to_numeric(s1, errors='coerce')
print(s2)
```

```
0       11
1       22
2    swift
3    30.82
4       33
dtype: object
Change the data type to numeric:
0    11.00
1    22.00
2      NaN
3    30.82
4    33.00
dtype: float64
```

In [ ]:

```
#Write a Pandas program to convert a given Series to an array.
import pandas as pd
import numpy as np
s1 = pd.Series(['17', '15', '52', '18.21', 'Paris'])
print(s1)
print("Converting to an array")
a = np.array(s1.values.tolist())
print (a)
```

```
0       17
1       15
2       52
3    18.21
4    Paris
dtype: object
Converting to an array
```

```
['17' '15' '52' '18.21' 'Paris']
```

In [ ]:

```
#Write a Pandas program to sort a given Series.
import pandas as pd
p = pd.Series(['Paris', '21', 'Tokyo', '27.21', '41'])
print(p)
new_p = pd.Series(p).sort_values()
print(new_p)
```

```
0     Paris
1        21
2     Tokyo
3     27.21
4        41
dtype: object
1        21
3     27.21
4        41
0     Paris
2     Tokyo
dtype: object
```

In [ ]:

```
#Write a Pandas program to create the mean and standard deviation of the data
import pandas as pd
s = pd.Series(data = [1,2,3,4,5,6])
print(s)
print("Mean:",s.mean())
print("Standard deviation:",s.std())
```

```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
Mean: 3.5
Standard deviation: 1.8708286933869707
```

In [ ]:

```
#Write a Pandas program to get the items of a given series not present in ano
import pandas as pd
sr1 = pd.Series([7, 2, 10, 4, 5])
sr2 = pd.Series([2, 4, 6, 8, 10])
print("\nItems of sr1 not present in sr2:")
result = sr1[~sr1.isin(sr2)]
print(result)
```

```
Items of sr1 not present in sr2:
0    7
4    5
dtype: int64
```

```
In [ ]:
#Write a Pandas program to get the items which are not common of two given se
import pandas as pd
import numpy as np
sr1 = pd.Series([1, 2, 3, 4, 5])
sr2 = pd.Series([2, 4, 6, 8, 10])
print("\nItems of a given series not present in another given series:")
sr11 = pd.Series(np.union1d(sr1, sr2))
sr22 = pd.Series(np.intersect1d(sr1, sr2))
result = sr11[~sr11.isin(sr22)]
print(result)
```

```
Items of a given series not present in another given series:
0     1
2     3
4     5
5     6
6     8
7    10
dtype: int64
```

```
In [ ]:
#Write a Pandas program to compute the minimum, 25th percentile, median, 75th
import pandas as pd
import numpy as np
num_state = np.random.RandomState(50)
num_series = pd.Series(num_state.normal(10, 4, 20))
print("Original Series:")
print(num_series)
result = np.percentile(num_series, q=[0, 25, 50, 75, 100])
print("\nMinimum, 25th percentile, median, 75th, and maximum of a given serie
print(result)
```

```
Original Series:
0      3.758592
1      9.876090
2      7.516286
3      4.141678
4     15.647784
5      8.093071
6      6.878123
7     14.281071
8      4.870830
9      4.690084
10    10.505351
11    13.448775
12    12.786948
13     8.661739
14     6.009896
15    16.395633
16    23.256301
17    13.951082
18    10.495465
19    12.971142
dtype: float64

Minimum, 25th percentile, median, 75th, and maximum of a given series:
[ 3.75859157  6.66106629 10.18577731 13.57435161 23.25630138]
```

In [ ]:

```
#Write a Pandas program to display most frequent value in a given series and 
import pandas as pd
import numpy as np
np.random.RandomState(100)
num_series = pd.Series(np.random.randint(1, 5, [15]))
print("Original Series:")
print(num_series)
print("Top 2 Freq:", num_series.value_counts())
result = num_series[~num_series.isin(num_series.value_counts().index[:1])] =
print(num_series)
```

```
Original Series:
0     2
1     2
2     1
3     4
4     2
5     2
6     3
7     1
8     1
9     3
10    3
11    4
12    1
13    1
14    4
dtype: int32
Top 2 Freq: 1    5
2    4
4    3
3    3
dtype: int64
0     Other
1     Other
2         1
3     Other
4     Other
5     Other
6     Other
7         1
8         1
9     Other
10    Other
11    Other
12        1
13        1
14    Other
dtype: object
```

In [ ]:

```
#Write a Pandas program to extract items at given positions of a given series
import pandas as pd
num_series = pd.Series(list('2390238923902390239023'))
element_pos = [0, 2, 6, 11, 21]
result = num_series.take(element_pos)
print("\nExtract items at given positions of the said series:")
print(result)
```

```
Extract items at given positions of the said series:
0     2
2     9
6     8
11    0
21    3
dtype: object
```

In [ ]:

```
#Write a Pandas program convert the first and last character of each word to
import pandas as pd
s = pd.Series(['london', 'mumbai', 'paris', 'madrid'])
result = s.map(lambda x: x[0].upper() + x[1:-1] + x[-1].upper())
print("First and last character of each word to upper case:")
print(result)
```

```
First and last character of each word to upper case:
0    LondoN
1    MumbaI
2     PariS
3    MadriD
dtype: object
```

In [ ]:

```
#Write a Pandas program to convert a series of date strings to a timeseries.
import pandas as pd
date_series = pd.Series(['02 Apr 2011', '22-06-2018', '20200307', '2021/05/06
print("Date strings to a timeseries:")
print(pd.to_datetime(date_series))
```

```
Date strings to a timeseries:
0   2011-04-02 00:00:00
1   2018-06-22 00:00:00
2   2020-03-07 00:00:00
3   2021-05-06 00:00:00
4   2010-04-12 00:00:00
5   2019-04-06 11:20:00
dtype: datetime64[ns]
```

In [ ]:

```
import pandas as pd
from dateutil.parser import parse
date_series = pd.Series(['01 Jan 2015', '10-02-2016', '20180307', '2014/05/06
print("Original Series:")
print(date_series)
date_series = date_series.map(lambda x: parse(x))
print("Day of month:")
print(date_series.dt.day.tolist())
print("Day of year:")
print(date_series.dt.dayofyear.tolist())
print("Week number:")
print(date_series.dt.weekofyear.tolist())
print("Day of week:")
print(date_series.dt.isocalendar().week.tolist())
```

```
Original Series:
0         01 Jan 2015
1          10-02-2016
2            20180307
3          2014/05/06
4          2016-04-12
5    2019-04-06T11:20
dtype: object
Day of month:
[1, 2, 7, 6, 12, 6]
Day of year:
[1, 276, 66, 126, 103, 96]
Week number:
[1, 39, 10, 19, 15, 14]
Day of week:
[1, 39, 10, 19, 15, 14]
```

```
C:\Users\deepdesk\AppData\Local\Temp/ipykernel_6104/1185217349.py:12: FutureWa
rning: Series.dt.weekofyear and Series.dt.week have been deprecated.  Please u
se Series.dt.isocalendar().week instead.
  print(date_series.dt.weekofyear.tolist())
```

In [ ]:

```
#Write a Pandas program to calculate the number of characters in each word in
import pandas as pd
series1 = pd.Series(['tokyo', 'mumbai', 'paris', 'rome'])
result = series1.map(lambda x: len(x))
print("Number of characters in each word in the said series:")
print(result)
```

```
Number of characters in each word in the said series:
0    5
1    6
2    5
3    4
dtype: int64
```

In [ ]:

```
#Write a Pandas program to get the powers of an array values element-wise.  No
import pandas as pd
df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72
print(df)
```

```
    X   Y   Z
```

```
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

In [ ]:

```python
#Write a Pandas program to create and display a DataFrame from a specified di
import pandas as pd
import numpy as np

exam_data  = {'Name': ['Ronaldo', 'Messi', 'Benzima', 'Pique', 'Ramos'],
        'Goals': [41, 40, 38, np.nan, 9,],
        'attempts': [55, 57, 56, 4, 20],
        'qualify': ['yes', 'yes', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e']

df = pd.DataFrame(exam_data , index=labels)
print(df)
```

```
      Name  Goals  attempts qualify
a  Ronaldo   41.0        55     yes
b    Messi   40.0        57     yes
c  Benzima   38.0        56     yes
d    Pique    NaN         4      no
e    Ramos    9.0        20      no
```

In [ ]:

```python
#Write a Pandas program to display a summary of the basic information about a
import pandas as pd
import numpy as np

exam_data  = {'Name': ['Ronaldo', 'Messi', 'Benzima', 'Pique', 'Ramos'],
        'Goals': [41, 40, 38, np.nan, 9,],
        'attempts': [55, 57, 56, 4, 20],
        'qualify': ['yes', 'yes', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e']

df = pd.DataFrame(exam_data , index=labels)
print("Summary of the basic information about this DataFrame and its data:")
print(df.info())
```

```
Summary of the basic information about this DataFrame and its data:
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, a to e
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      5 non-null      object 
 1   Goals     4 non-null      float64
 2   attempts  5 non-null      int64  
 3   qualify   5 non-null      object 
dtypes: float64(1), int64(1), object(2)
memory usage: 200.0+ bytes
None
```

In [ ]:

```
import pandas as pd
import numpy as np

exam_data  = {'Name': ['Ronaldo', 'Messi', 'Benzima', 'Pique', 'Ramos'],
        'Goals': [41, 40, 38, np.nan, 9,],
        'attempts': [55, 57, 56, 4, 20],
        'qualify': ['yes', 'yes', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e']
df = pd.DataFrame(exam_data , index=labels)
print("First three rows of the data frame:")
print(df.iloc[:3])
```

```
First three rows of the data frame:
      Name  Goals  attempts qualify
a  Ronaldo   41.0        55     yes
b    Messi   40.0        57     yes
c  Benzima   38.0        56     yes
```

In [ ]:

```
import pandas as pd
import numpy as np

exam_data  = {'Name': ['Ronaldo', 'Messi', 'Benzima', 'Pique', 'Ramos'],
        'Goals': [41, 40, 38, np.nan, 9,],
        'attempts': [55, 57, 56, 4, 20],
        'qualify': ['yes', 'yes', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e']

df = pd.DataFrame(exam_data , index=labels)
print("Select specific columns and rows:")
print(df.iloc[[1, 3, 4,], [1, 3]])
```

```
Select specific columns and rows:
   Goals qualify
b   40.0     yes
d    NaN      no
e    9.0      no
```

In [ ]:

```
#Write a Pandas program to count the number of rows and columns of a DataFrame
import pandas as pd
import numpy as np

exam_data  = {'Name': ['Ronaldo', 'Messi', 'Benzima', 'Pique', 'Ramos'],
        'Goals': [41, 40, 38, np.nan, 9,],
        'attempts': [55, 57, 56, 4, 20],
        'qualify': ['yes', 'yes', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e']

df = pd.DataFrame(exam_data , index=labels)
total_rows=len(df.axes[0])
total_cols=len(df.axes[1])
print("Number of Rows: "+str(total_rows))
print("Number of Columns: "+str(total_cols))
```

```
Number of Rows: 5
Number of Columns: 4
```

In [ ]:

```
#Write a Pandas program to count the number of rows and columns of a DataFrame
import pandas as pd
import numpy as np

exam_data  = {'Name': ['Ronaldo', 'Messi', 'Benzima', 'Pique', 'Ramos'],
        'Goals': [41, 40, 38, np.nan, 9,],
        'attempts': [55, 57, 56, 4, 20],
        'qualify': ['yes', 'yes', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e']

df = pd.DataFrame(exam_data , index=labels)
total_rows=len(df.axes[0])
total_cols=len(df.axes[1])
print("Number of Rows: "+str(total_rows))
print("Number of Columns: "+str(total_cols))
```

```
Number of Rows: 5
Number of Columns: 4
```

In [ ]:

```
#Write a Pandas program to select the rows where number of attempts in the exa
#Sample Python dictionary data and list labels:
#exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'M
#'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
#'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
#'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']
#labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', labels = ['a', 'b', 'c', 'd', '
#Expected Output:
#Number of attempts in the examination is less than 2 and score greater than
#name score attempts qualify
#j Jonas 19.0 1 yes
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'M
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Number of attempts in the examination is less than 2 and score greater
print(df[(df['attempts'] < 2) & (df['score'] > 15)])
```

```
Number of attempts in the examination is less than 2 and score greater than 15
:
    name  score  attempts qualify
j  Jonas   19.0         1     yes
```

```
In [ ]:
#Write a Pandas program to calculate the sum of the examination attempts by t
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'M
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nSum of the examination attempts by the students:")
print(df['attempts'].sum())
```

```
Sum of the examination attempts by the students:
19
```

```
In [ ]:
#Write a Pandas program to calculate the mean score for each different studen
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'M
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nMean score for each different student in data frame:")
print(df['score'].mean())
```

```
Mean score for each different student in data frame:
13.5625
```

```
In [ ]:
#Write a Pandas program to append a new row 'k' to data frame with given valu
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'M
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Original rows:")
print(df)
print("\nAppend a new row:")
df.loc['k'] = [1, 'Suresh', 'yes', 15.5]
print("Print all records after insert a new record:")
print(df)
print("\nDelete the new row and display the original  rows:")
df = df.drop('k')
print(df)
```

```
Original rows:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

```
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes

Append a new row:
Print all records after insert a new record:
        name   score attempts qualify
a  Anastasia    12.5        1     yes
b       Dima     9.0        3      no
c  Katherine    16.5        2     yes
d      James     NaN        3      no
e      Emily     9.0        2      no
f    Michael    20.0        3     yes
g    Matthew    14.5        1     yes
h      Laura     NaN        1      no
i      Kevin     8.0        2      no
j      Jonas    19.0        1     yes
k          1  Suresh      yes    15.5

Delete the new row and display the original  rows:
        name score attempts qualify
a  Anastasia  12.5        1     yes
b       Dima   9.0        3      no
c  Katherine  16.5        2     yes
d      James   NaN        3      no
e      Emily   9.0        2      no
f    Michael  20.0        3     yes
g    Matthew  14.5        1     yes
h      Laura   NaN        1      no
i      Kevin   8.0        2      no
j      Jonas  19.0        1     yes
```

In [ ]:

```
#Write a Pandas program to sort the DataFrame first by 'name' in descending o
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Mi
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
df.sort_values(by=['name', 'score'], ascending=[False, True])
print(df)
```

```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

In [ ]:

```
#Write a Pandas program to replace the 'qualify' column contains the values 'y
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Mi
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Replace the 'qualify' column contains the values 'yes' and 'no'  with T
df['qualify'] = df['qualify'].map({'yes': True, 'no': False})
print(df)
```

```
Replace the 'qualify' column contains the values 'yes' and 'no'  with True and
False:
        name  score  attempts  qualify
a  Anastasia   12.5         1     True
b       Dima    9.0         3    False
c  Katherine   16.5         2     True
d      James    NaN         3    False
e      Emily    9.0         2    False
f    Michael   20.0         3     True
g    Matthew   14.5         1     True
h      Laura    NaN         1    False
i      Kevin    8.0         2    False
j      Jonas   19.0         1     True
```

In [ ]:

```
#Write a Pandas program to change the name 'James' to 'Suresh' in name column
import pandas as pd
import numpy as np
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'M
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Change the name 'James' to ‘Suresh’:")
df['name'] = df['name'].replace('James', 'Suresh')
print(df)
```

```
Change the name 'James' to ‘Suresh’:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d     Suresh    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

In [ ]:

```
#Write a Pandas program to iterate over rows in a DataFrame.
import pandas as pd
import numpy as np
exam_data = [{'name':'tim', 'score':12.5}, {'name':'joe','score':9}, {'name':
df = pd.DataFrame(exam_data)
for index, row in df.iterrows():
    print(row['name'], row['score'])
```

```
tim 12.5
joe 9.0
rachel 16.5
```

In [ ]:

```
import pandas as pd
import numpy as np
d = {'name': ['Ronaldo','Messi','Benzima'], 'c_clubs': ['Manchester United','
df = pd.DataFrame(data=d)
print('After add one row:')
df2 = {'name':'Ramos', 'c_clubs':'PSG', 'p_clubs':'Real Madrid'}
df = df.append(df2, ignore_index=True)
print(df)
```

```
After add one row:
      name            c_clubs      p_clubs
0  Ronaldo  Manchester United     Juventus
1    Messi                PSG    Barcelona
2  Benzima        Real Madrid          N/A
3    Ramos                PSG  Real Madrid
```

In [ ]:

```
import pandas as pd
import numpy as np
d = {'name': ['Ronaldo','Messi','Benzima'], 'c_clubs': ['Manchester United','
df = pd.DataFrame(data=d)
print('After altering name and p_clubs')
df = df[['p_clubs', 'c_clubs', 'name']]
print(df)
```

```
After altering name and p_clubs
     p_clubs            c_clubs     name
0   Juventus  Manchester United  Ronaldo
1  Barcelona                PSG    Messi
2        N/A        Real Madrid  Benzima
```

In [ ]:

```
#Write a Pandas program to select rows from a given DataFrame based on values
import pandas as pd
import numpy as np
d = {'name': ['Ronaldo','Messi','Benzima'], 'c_clubs': ['Manchester United','
df = pd.DataFrame(data=d)
print('Rows for name value == Messi')
print(df.loc[df['name'] == 'Messi'])
```

```
Rows for name value == Messi
    name c_clubs    p_clubs
1  Messi     PSG  Barcelona
```

In [ ]:

```
#Write a Pandas program to rename columns of a given DataFrame
import pandas as pd
import numpy as np
d = {'name': ['Ronaldo','Messi','Benzima'], 'c_clubs': ['Manchester United','
df = pd.DataFrame(data=d)
df.columns = ['Column1', 'Column2', 'Column3']
df = df.rename(columns={'name': 'Column1', 'c_clubs': 'Column2', 'p_clubs': '
print(df)
```

```
   Column1            Column2    Column3
0  Ronaldo  Manchester United   Juventus
1    Messi                PSG  Barcelona
2  Benzima        Real Madrid        N/A
```

In [ ]:

```
#Write a Pandas program to get list from DataFrame column headers.
import pandas as pd
import numpy as np
d = {'name': ['Ronaldo','Messi','Benzima'], 'c_clubs': ['Manchester United','
df = pd.DataFrame(data=d)
print(list(df.columns.values))
```

```
['name', 'c_clubs', 'p_clubs']
```