1 - Get me list of odd numbers between 1 to 20 without using if condition.

```
for i in range(1,21,2):
   print(i)
```

```
1
3
5
7
9
11
13
15
17
19
```

2 - Get a list of 1 to 20 then remove elements from list to get only even elements.

```
l1 = [i for i in range(1,21,1)]
print(l1)
l2 = [j for j in l1 if j%2==0]
print(l2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

3 - Get a list of 1 to 8 and then 4 to 10. Get the common elements from both the list in a new list.

```
l1 = [i for i in range(1,9)]
print(l1)
l2 = [i for i in range(4,11)]
print(l2)
l3 = []
for i in range(len(l1)):
  for j in range(len(l2)):
    if l1[i] == l2[j]:
      l3.append(l1[i])
print(l3)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
[4, 5, 6, 7, 8, 9, 10]
[4, 5, 6, 7, 8]
```

4 - Sort a shuffled list of 10 random numbers in descending order.

```
import random
rl = random.sample(range(1, 1000), 10)
print(rl)
```

```
rl.sort(reverse=True)
print(rl)
```

```
[626, 110, 998, 851, 842, 922, 760, 990, 424, 660]
[998, 990, 922, 851, 842, 760, 660, 626, 424, 110]
```

5 - x=(1,2,3,4,5), y=(4,5,6,7). Combine these two tuples in a single tuple ignoring the common elements.

```
x = (1,2,3,4,5)
y = (4,5,6,7)
u = tuple(set(x + y))
u
```

```
(1, 2, 3, 4, 5, 6, 7)
```

6 - Define two sets and perform all the set operations and validation operations.

```
a = {2,3,5}
b = {9,8,7,2,3,5,19}
print("Original set a : ",a)
print("Original set b : ",b)
c = b.copy()
print("Set c(Copy of b) : ",c)
c.add(19)
c.remove(9  )
print("After add : ",c)
c.discard(5)
print("Discard of 5 from : ",c)
print("Difference of a and b : ",a.difference(b))
print("Intersection of a and b : ",a.intersection(b))
print("Is a subset of b : ",a.issubset(b))
print("Is b superset of a : ",b.issuperset(a))
a.clear()
```

```
Original set a :  {2, 3, 5}
Original set b :  {2, 3, 5, 7, 8, 9, 19}
Set c(Copy of b) :  {2, 3, 19, 5, 7, 8, 9}
After add :  {2, 3, 19, 5, 7, 8}
Discard of 5 from :  {2, 3, 19, 7, 8}
Difference of a and b :  set()
Intersection of a and b :  {2, 3, 5}
Is a subset of b :  True
Is b superset of a :  True
```

7 - Generate a dictionary {1:1,2:1,3:1,4:1,...,10:1} in one line using dictionary's method.

```
d1 = dict.fromkeys((1,2,3,4,5,6,7,8,9,10),1)
d1
```

```
{1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 1, 9: 1, 10: 1}
```

8 - Print all the keys and values of a dictionary.

```
print(d1.keys())
print(d1.values())
```

```
dict_keys([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
dict_values([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

9 - Two dictionaries {'a':1,'b':2,'c':3}, {'a':4,'d':5,'e':6}. Merge these two dictionaries.

```
d1 = {'a': 1, 'b': 2, 'c': 3}
d2 = {'a': 4, 'd': 5, 'e': 6}

def merge(d1, d2):
   d3 = {**d1, **d2}
   for key, value in d3.items():
       if key in d1 and key in d2:
               d3[key] = [value , d1[key]]
   return d3

d3 = merge(d1, d2)
print(d3)
```

```
{'a': [4, 1], 'b': 2, 'c': 3, 'd': 5, 'e': 6}
```

10 - How to check whether a key is existing in a dictionary or not.

```
d1 = {'a': 1, 'b': 2, 'c': 3}
if d1.get('a') == None:
  print("Key not present")
else:
  print("Key Exists")
```

```
Key Exists
```

11 - How can we have two variables refering to a single list, set and dictionary.

```
a = [1, 2, 3]
b = a
print(b)

a = {1, 2, 3}
b = a
print(b)

a = {'a': 1, 'b': 2}
b = a
print(b)
```

```
[1, 2, 3]
{1, 2, 3}
{'a': 1, 'b': 2}
```

12 - Use all the case methods of strings.

```
s = "Hello, World!"

print(s.upper())
print(s.lower())
print(s.title())
print(s.capitalize())
print(s.swapcase())
```

```
HELLO, WORLD!
hello, world!
Hello, World!
Hello, world!
hELLO, wORLD!
```

13 - How to split a string with a substring?

```
s = "Hello, World!"
substrings = s.split(",")
print(substrings)
```

```
['Hello', ' World!']
```

14 - How to replace a string with a substring?

```
s = "Hello, World!"
s1 = s.replace("World", "Earth")
print(s1)
```

```
Hello, Earth!
```

15 - How to join multiple strings with a substring?

```
s = ["Hello", "World", "How", "Are", "You"]
s1 = ", ".join(s)
print(s1)
```

```
Hello, World, How, Are, You
```

16 - How to make partition of a string?

```
s = "Hello, World!"
s1 = s.partition(",")
print(s1)
```

```
('Hello', ',', ' World!')
```

17 - Get the last element of the list, tuple and string.

```
l = [1, 2, 3, 4, 5]
last = l[-1]
print("List's Last element : ",last)
t = (1, 2, 3, 4)
last = t[-1]
print("List's Last element : ",last)
s = "hello"
last = s[-1]
print("List's Last element : ",last)
```

```
List's Last element :  5
List's Last element :  4
List's Last element :  o
```

18 - Get last 3 elements of the list, tuple and string.

```
l = [1, 2, 3, 4, 5]
last = l[-3:]
print("List's Last three element : ",last)
t = (1, 2, 3, 4)
last = t[-3:]
print("List's Last three element : ",last)
s = "hello"
last = s[-3:]
print("List's Last three element : ",last)
```

```
List's Last three element :  [3, 4, 5]
List's Last three element :  (2, 3, 4)
List's Last three element :  llo
```

19 - Get first 5 elements of list, tuple and string.

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
first_five = lst[:5]
print(first_five)
tpl = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
first_five = tpl[:5]
print(first_five)
s = "Hello, World"
first_five = s[:5]
print(first_five)
```

```
[1, 2, 3, 4, 5]
(1, 2, 3, 4, 5)
Hello
```

20 - How to find the no of occurences of a substring?

```
s = "Hello, World! How are you?"
occurrences = s.count("o")
print(occurrences)
```

```
4
```

21 - Use all the validation methods used with string.

```
s1 = "Hello"
s2 = "Hello123"
s3 = "123"
s4 = "hello"
s5 = "HELLO"
s6 = "Hello World"

print(s1.isalpha())
print(s2.isalpha())
print(s3.isalpha())

print(s1.isalnum())
print(s2.isalnum())
print(s3.isalnum())

print(s1.isdigit())
print(s2.isdigit())
print(s3.isdigit())

print(s1.islower())
print(s4.islower())

print(s1.isupper())
print(s5.isupper())
```

```
print(s1.istitle())
print(s6.istitle())
```

```
True
False
False
True
True
True
False
False
True
False
True
False
True
True
True
```

22 - Convert all the data structures to other data structures.

```
lst = [1, 2, 3, 4]
tpl = tuple(lst)
print(type(tpl))
print(tpl)
l = list(tpl)
print(type(l))
print(l)
s = set(l)
print(type(s))
print(s)
lst = list(s)
print(type(lst))
print(lst)
d = {1: "a", 2: "b", 3: "c"}
print(type(d))
lst = list(d.items())
print(type(lst))
print(lst)
```

```
<class 'tuple'>
(1, 2, 3, 4)
<class 'list'>
[1, 2, 3, 4]
<class 'set'>
{1, 2, 3, 4}
<class 'list'>
[1, 2, 3, 4]
<class 'dict'>
<class 'list'>
[(1, 'a'), (2, 'b'), (3, 'c')]
```

23 - Get all the elements excluding first and last elements from list, tuple and string.

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(lst[1:-1])
tpl = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print(tpl[1:-1])
s = "Hello, World! How are you?"
print(s[1:-1])
```

```
[2, 3, 4, 5, 6, 7, 8, 9]
(2, 3, 4, 5, 6, 7, 8, 9)
ello, World! How are you
```

24 - Get all the elements in a list using : operator.

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(lst[::])
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

25 - Get last 5 elements from a list of 1 to 10 using negative indexing.

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
last_five = lst[-5:]
print(last_five)
```

```
[6, 7, 8, 9, 10]
```

26 - Get 4 elements of the list excluding last 2 elements using negative indexing.

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
four = lst[4:-2]
print(four)
```

```
[5, 6, 7, 8]
```

27 - Convert a list of tuple to dictionary.

```
lst = [('a', 1), ('b', 2), ('c', 3)]
dct = dict(lst)
print(dct)
```

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

28 - Create a dictionary using range() as following. {'a':1, 'b':2, 'c':3, 'd':4, 'e':5. 'y':25,'z':26}. The code needs to be in one line.

```
import string
dict(zip(string.ascii_lowercase,range(1,27)))
```

```
{'a': 1,
 'b': 2,
 'c': 3,
 'd': 4,
 'e': 5,
 'f': 6,
 'g': 7,
 'h': 8,
 'i': 9,
 'j': 10,
 'k': 11,
 'l': 12,
 'm': 13,
 'n': 14,
 'o': 15,
 'p': 16,
 'q': 17,
 'r': 18,
 's': 19,
 't': 20,
 'u': 21,
 'v': 22,
 'w': 23,
 'x': 24,
 'y': 25,
 'z': 26}
```

29 - There are two lists [1,2,3,4,5,6,7,8,9,10],[11,12,13,14,15,16,17,18,19,20]. Get a third list from these two lists as [12,14,16,18,20,22,24,26,28,30].

```
lst1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
lst2 = [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
lst3 = [x + y for x, y in zip(lst1, lst2)]
print(lst3)
```

```
[12, 14, 16, 18, 20, 22, 24, 26, 28, 30]
```

30 - Get Square of all the elments in a list from 1 to 10 numbers.

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
squared_lst = [x**2 for x in lst]
print(squared_lst)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

31 - There are two lists [1,2,3,4,5], [4,5,6,7] get a list from these two lists [1,2,3,6,7].

```
a = [1, 2, 3, 4, 5]
b = [4, 5, 6, 7]
for i in a[:]:
    if i in b:
        a.remove(i)
        b.remove(i)
c = a+b
c
```

```
[1, 2, 3, 6, 7]
```

32 - Create a function for string that will check whether a string is having the first letter as Capital and not anyother letter is capital.

```
def checkString(s):
  if s.istitle() == True:
    print("String is in format")
  else:
    print("String not in format")

checkString("Hello")
```

```
String is in format
```

33 - Create a class Student and add member variables with False values. The variables are as listed below. Marks will have a default value blank list.

1. Name
2. Reg No
3. Roll No
4. Standard
5. Admission Year
6. Marks
7. Result

34 - Add a constructor that will assign Name, Reg No, Roll No, Standard, Admission year. In the constructor add validation for Name to be only Alphabetic, Reg No to be alphanumeric, Roll No, Standard nad Admission year to be numeric. All abobve values will be accepted as string only.

35 - Add a method that will accept a dictionary for marks containing subject as key and marks as values. It will add the dictionary to the list of marks. Marks list will have multiple elements and each element will be a dictionary only. Here there should be a validation to acccept the marks which are less than or equal to 100 only. If the obtained marks are less than 40 the result willl be fail otherwise pass. In the dictionary the result can be added.

36 - Add another method that will generate the result. This method will check if there is any line in the marks having fail as result the result will be Fail and it will print the complete result as following. Name : Roll No : Standard:

---

37 Subject Total MarksPassing MarksObtained MarksResult <Sub 1> 100 40 <Sub 1> 100 40 <Sub 1> 100 40
TOTAL

Result: PASS / FAIL Percentage:

```
class Student:
    name = ""
    reg_no = ""
    roll_no = ""
    standard = ""
    admission_year = ""
    marks = []
    result = False
    def __init__(self, name, reg_no, roll_no, standard, admission_year):
        if not name.isalpha():
            raise ValueError("Name must be alphabetic")
        if not reg_no.isalnum():
            raise ValueError("Reg No must be alphanumeric")
        if not roll_no.isdigit():
            raise ValueError("Roll No must be numeric")
        if not standard.isdigit():
            raise ValueError("Standard must be numeric")
        if not admission_year.isdigit():
            raise ValueError("Admission year must be numeric")

        self.name = name
        self.reg_no = reg_no
        self.roll_no = roll_no
        self.standard = standard
        self.admission_year = admission_year
        self.marks = []
        self.result = False

    def add_marks(self, marks_dict):
        if not all(0 <= value <= 100 for value in marks_dict.values()):
            raise ValueError("Marks must be between 0 and 100")

        self.marks.append(marks_dict)

        if all(value >= 40 for value in marks_dict.values()):
            self.result = True
        else:
            self.result = False
    def generate_result(self):
        rs = ""
```

```
        if any(mark.get("result", False) == False for mark in self.marks):
            rs = "Fail"
        else:
            rs = "Pass"
        print(f"Name: {self.name} Roll No: {self.roll_no} Standard: {self.standard}
        print("Subject\tTotal Marks\tPassing Marks\tObtainedMarks\tResult")
        s = 0
        t = 0
        p = 0
        r = ""
        for key,value in self.marks[0].items():
          if value>=40:
            r = "Pass"
          else:
            r = "Fail"
          print(f"{key} \t100\t40\t{value}\t{r}")
          s +=value
          t +=100
          p +=40
        print(f"TOTAL \t{t}\t{p}\t{s}\t{rs}")

student1 = Student("John", "123456", "1", "10", "2021")
student1.add_marks({"Math": 85, "Science": 90, "English": 95})
student1.generate_result()
```

```
Name: John Roll No: 1 Standard: 10
Subject Total Marks     Passing Marks   ObtainedMarks   Result
Math    100     40      85      Pass
Science         100     40      90      Pass
English         100     40      95      Pass
TOTAL   300     120     270     Fail
```