To Reverse number:

```
p=134
rev=0
for
while p>0:
    a=p%10
    rev=rev*10+a
    p=p//10
print(rev)
```

To remove duplicates:

```
p=[1, 2, 3, 2, 4, 3, 5]
l=[]
for i in p:
    if i not in l:
        l.append(i)
print(l)
```

⇒ To Count no of element

```
for i in q:
    print(i, p.count(i))
```

To revesse string

```
p='sachin'
i=len(p)-1  sts=[]
while i>=0:
    sts.append(p[i])  # imp
    i=i-1
print(''.join(sts))
```

Identity Matrix

```
l=[]
n=int(input('enter the num:'))
for row in range(n):
    l1=[]
    for col in range(n):
        if row==col:
            l1.append(1)
        else:
            l1.append(0)
    l.append(l1)
```

To Count no of element by using dict

```
p=[1,2,3,2,5,4,8]
dict={}
for i in p:
    if i not in dict:
        dict[i]=1
    else:
        dict[i]=dict[i]+1
print(dict)
for k,v in dict.items():
    print(k,v)
```

To Find ovels in string:

```
p='python is very easy language'
q=[]
for i in p:
    if i in('a','e','i','o','u'):
        q.append(i)
print(q)
```

To combine 2 dictionaries by adding values of the common keys

```
d1 = {'a':200, 'b':200 'c':100}
d2 = {'a':200, 'b':100 'd':100}

for k in d2:
    for i in d1:
        if k == i:
            d1[i] = d2[k] + d1[i]

print(d1)
```

{'a':400, 'b':300}

Sorted (alphabate), sorted(digit)

S = 'B4A1D3' ⟹ 'ABD134'

```
alphabate = []
digit = []

for ch in S:
    if ch.isalpha():
        alphabate.append(ch)
    else:
        digit.append(ch)

output = ''.join(sorted(alphabate) + sorted(digit))

print(output)
```

---

p = 'a4z2n3'

o/p = 'aaaazznnn'

```
output = ''

for i in p:
    if i.isalpha():
        x = i
    else:
        d = int(i)
        output = output + x*d

print(output)
print(''.join(sorted(output)))
```

S = 'aaabbccz'

o/p ⇒ a3b2c2z1

```
previous = S[0]
c = 1
i = 1
output = ''
while i < len(S):
    if S[i] == previous
        c = c+1
    else:
        output = output + str(c) + previ
        previous = S[i]
        c = 1
    if i == len(S)-1:
        output = output + str(c)
            previous
    i = i+1
print(output)
```

To print 'n' Fibonacci number

```
def fib(n):
    a, b = 0, 1
    if n == 1:
        print(a)
    else:
        print(a)
        print(b)
        for i in range(2, n):
            c = a+b
            a = b
            b = c
            print(c)
```

or

```
num = int(input('enter the no'))
a = 0
b = 1
c = 0
while c < = num:
    print(c)
    a = b
    b = c
    c = a+b
```

without using third variable

```
a, b = 0, 1
while b < 50:
    print(b)
    a, b = b, a+b
```

To print prime number

```
for i in range(50)
    if i > 1:
        for j in range(2, i):
            if i%2 == 0:
                Break
        else:
            print(i)
```

To find max length word in string

```
p = 'python is very easy
     programming language'
q = p.split()
c = 0
i = 0
s = ''
for i in q:
    if len(i) > c:
        c = len(i)
        s = i
print(s)
```

```
s = 'ABAABBCA'
o/p = A4B3C1
dict = {}
output = ''
for i in s:
    if i not in dict:
        dict[i] = 1
    else:
        dict[i] = dict[i] + 1
for K,V in dict.items():
    output = output + K + str(V)
print(output)
```

Check two strings Anagram or not

```
s1 = 'lazy'  s2 = 'zaly'
if sorted(s1) == sorted(s2)
    print('both strings are Anagram')
```

---

to check pelindrome or not

```
s1 = eye
if s1 == s1[::-1]
    print('its pelindrome')
```

To replace ovels with '$'

```
p = 'sachin'
for i in p:
    if i in ('a','i','e','o','u'):
        q = p.replace(i,'$')
print(q)
```

Threading :-

```
def display():
    for i in range(5):
        print('in child class')

t = threading.Thread(Target=display)
t.start()

for i in range(5):
    print('in main thread')
```

Exception :- unaccepted result during execution

```
Try:
    a = int(input('enter the num'))
    b = int(input('enter the num'))
    c = a/b
except:
    print('zerodevision error')
else:
    print('c')
finally:
    print('in finally')
```

---

(1) enumerate(iterable, start=0)
↳ take one iterable

(2) zip(iterables)
↳ combines respective elements of iterables

(3) map(function, iterables)
↳ maps function with iterables

(4) filter(function, iterable)
↳

(5) r = reduce(func^n, iterable,)
from functools import reduce
→ we get composite result like sum
r = reduce(lambda x,y: x+y, p)

List Comprehension:

It is consize way to declare a list, where each element is the result of some expression

```
p=[1,2,3,4,5,6]
l=[x for x in p if x%2==0]
```

dict. Comprehension:-

```
p=[1,2,3,4]
d={x:x**2 for x in p
    if x%2==0}
print(d) ⇒ {2:4, 4:8}
d={'a':2,'b':4}
d={k:v*2 for k,v in dict.items()}
```

Generator Comprehension:

```
p=[1,2,3,4,5]
g=(x for x in p if x%2==0)
for i in g:
    print(i)
```

Recursion:- A function calling itself is called Recursion

Factorial of number

```
def fact(n):
    if n==0 or n==1:
        return 1
    else
    return n * fact(n-1)
```

decorator:-

enhance the functionality of existing function without changing its original structure

```
def decorator(function):
    def inner():
        x=function()
        return x.upper()
    return inner

def name():
    return 'sachin'

name = decorator(name)
print(name())
```

Create dict with using two variable in class

```
class A:
    def __init__(self, name, age):
        self.age=age
        self.name=name

obj=A('sachin',32)
print(obj.__dict__)
```

WAP Key will be no and Value will be the list of 10 multiplier

```
d={}
for i in range(1,21):
    L=[]
    for j in range(1,11):
        L.append([i*j])
    d[i]=L
print(d)
```

---

wap to combine 2 dictionaries by adding values of the common keys

```
d1={'a':100,'b':200}
d2={'a':300,'c':100}

for i in d2:
    if i not in d1:
        d1[i]=d2[i]
    else:
        d1[i]=d2[i]+d1[i]

print(d1)
```

$d_1 = \{'a':400, 'b':200, 'c':100\}$

To Convert Key ⟹ Value and Value ⟹ Key

```
dict={'a':1,'b':2,'c':3}
dict={Value:Key for Key,value in dict.items()}
print(dict)
```

---

addition of numbers using lambda

```
p=[1,2,3,4,5,6]
from functools import reduce
l=reduce(lambda x,y:x+y,p)
print(l)
```

```
p=[1,2,3,4,5,6,7]
max=p[0]
for i in p:
    if i>max:
        max=i
print(max)
```

To Find largest Numbers in list

```
for i in p:
    if i<min:
        min=i
print(min)
```

---

addition of two numbers

Lambda

```
a=lambda x,y:x+y
x=int(input('enter the number'))
y=int(input('enter the number'))
print(a(x,y))
```

⟹ print(a(5,6))

---

shallow copy & deepcopy

```
import copy
p=[[1,2],[3,4]]
cp=copy.deepcopy(p)
```

deepcopy (don't want to change original copy)

```
sh=copy.copy(p)
```

shallow copy

New copy with existing Refrence

① Select *, newSal = (salary - (Salary/30)*2) from employee;

② with result as (using Dense rank)
(select *, Dense_rank() over (order by salary desc) as new_col
from employee) selet * from result where new_col = 2

③ Using TOP :-
Select min(salary) from employee where salary in
(selet Top 2 salary from Employee order by
Salary desc);

④ Select max(salary) from employee where salary <
(selet max(salary) from employee);

⑤ duplicate :- column name
Select name, salary, dept, count(*) from employee.
Group by name, salary, dept having count(*) > 1;
column name

⑥ Delete Duplicate :-
with result as
(select *, row_number() over (order by salary) as
new_col from employee) Delet from Result
where new_col > 1;

⑦ select destinct(dept) from employee;

⑧ Select dept, count(dept) from employee Group by dept;

⑨ select * from A inner Join B on A.AID = B.BID;

```
p=('a','e','i','o','u')
p1=[]
p2=[]
for i in p:
    if i in ('a','e',i):
        p1.append(i)
    else:
        p2.append(i)
print(tuple(p1))
print(tuple(p2))
```

('a','e',i) & ('o','u')

or

```
print(p[0:3])
print(p[3:6])
```

```
l=['a','b','c']
a=','
l1=[]
l1.append(a.join(l))
```

→ 'a,b,c'

Abstract method:

it has only declaration no implementation is called abstract method

```
from abc import ABC,
                abstractmethod.

class Aclass(ABC):
    @abstractmethod
    def show(self):
        pass

class Bclass(AClass):
    def show(self):
        print('defining abstract
                    class')
```