tuples have only two function

① count

② Index

* why are you use set?
→ set is gives the unique Data.

ex.

t=

ex. [2, 3, 4, 5, 'sudh', 45.56, false, 45+45j]

t.count(3)

↳ 1

t.Index('sudh')

↳ 4

ex. S4 = {2, 3, 4, 5, 4, 5, 2, 8}

↳ {2, 3, 4, 5, 8}

S5

↳ {12, 2, 23, 234, 342, 45, 456, 5

S5.add(4)

↳ {12, 2, 23, 4, 234, 342, 45

'abc', 789}

Set It will build an unordered collection of Unique element.

Set :-

A python set is the collection of the unordered items. each elements in the set must be unique, Immutable and the sets remove the duplicate elements.

→ sets are mutable which means we can modify it after its creation.

→ we cannot directly access any element of the set by the Index.

S1 = { }

type(S1)

↳ dict

S2 = {2, 3, 4, 55, 6}

type(S2)

↳ set

S3 = {324, 456, 456, "sudh", 45+45j, 34.465, [2, 3, 4]}

↳ typer error :- unhashable type: 'List'.

S3 = {324, 456, 456, "sudh", 45+45j, 34.465}

↳ {324, 'sudh', 456, 34.465, 45+45j}

Comprehension Function :- ✓

there are two types of compre

① List comprehension

② set comprehension :-

→ comprehension It is meth reduce more Line of code F of Code.

① ex. # L = [1, 2, 3, 4, 45, 5]

L1 = [ ]

for i in L1

L1.append(i**2)

L1

↳ [1, 4, 9, 16, 2025, 25]

② # L

↳ [1, 2, 3, 4, 45, 5]

[i**2 for i in L1)

List comprehension Function of

↳ [1, 4, 9, 16, 2025, 25]

② # statement :- Find the even numbe given a List. ex. Of List L.

[i for i in L if i%=

↳ [2, 4]

Print function always return Nontype
Nontype Datatype

M T W T F S S
Page No.:
Date:
YOUVA

④ ex. Find upper string in given List

L1 = ["sudh", "pwskill", "kumar", "Data science masters"]

[i.upper() for i in L1]

↳ ['SUDH', 'PWSKILL', 'KUMAR', 'DATA SCIENCE MASTERS']

**** ⑤ Tuples Comprehension :-

→ comprehension is work of tuple But they will generator object.

→ So you will Need to convert generator object into either List or for Loop to extract the data in generator object.

ex. # L =

↳ [1, 2, 3, 4, 5, 45, 5]

(i**2 for i in L)

↳ generator object

so use List or for Loop # List (i**2 for i in L)

↳ [1, 4, 9, 16, 2025, 25]

**** Dictionary Comprehension :-

ex. d = { "key1": 1, "key2": 2, "key3": 3, "key4": 4 }

for k,v in d.items()

d.items()

↳ dict_items([('key1', 1), ('key2', 2), ('key3', 3), ('key4', 4)])

② # { k : v**2 for k,v in d.items() }

↳ { 'key1': 1, 'key2': 4, 'key3': 9, 'key4': 16 }

statement :- Find the value grater than 1 in Dictionary d.

d

↳ { 'key1': 1, "key2": 2, "key3": 3, "key4": 4 }

{ k : v for k,v in d.items() if v>1 }

↳ { 'key2': 2, 'key3': 3, 'key4': 4 }

Function part 1 :-

* In Built function :-

print("this is my print")

↳ this is my print

L = [324, 45, 45, 45]

Len (L)

↳ 4

type (1) ↳ List

**** User defined function.

# def test1():
    print("this is my very very First Function")

↳ # test1()

↳ this is my very very First function

Note : return gives any datatype you call.

def test2():
    return "this is my very First return"

test2()

↳ this is my very first return

```
# test2() + "sudh"
↳ 'this is my very first return sudh'
```

```
# def test3():
    return 'sudh', 23, 345.56, [1,2,3,3]
# test3()
↳ ('sudh', 23, 345.56, [1,2,3,3])
# a,b,c,d = test3()
```

# a	# b	# c	# d
↳ 'sudh'	↳ 23	↳ 345.56	↳ [1,2,3,3]

```
# def test4():
    a = 5+6/7
    return a
# test4()
↳ 5.857142....
```

```
# def test5(a,b,c):
    d = a+b/c
    return d
# test5(2,5,8)
↳ 2.625
```

Statement :- Find Integer Numbers in given string.

```
# L = [1,2,3,4, "sudh", "Kumar", [1,2,3,4,5,6]]
```

Normal way to find.

```
# L1=[]
  for i in L:
    if type(i)==int or type(i)==float:
      L1.append(i)
# L1
[1,2,3,4]
```

Using def

```
# def test7(l):
    L1=[]
    for i in L:
      if type(i)==int or type(i)==float:
        L1.append(i)
    return L1
# test7(L)
↳ [1,2,3,4]
```

Statement :- Find the ~~Nu~~ Integer Number in given List inside the List.

```
# def test8(a):
    L1=[]
    for i in a:
      if type(i)==List:
        for j in i
          L1.append(j)
      else:
        if type(i)==int or type(i)==float:
          L1.append(i)
    return L1
# test8(L)
↳ [1,2,3,4,1,2,3,4,5,6]
```

Old List `# L`

```
↳ [1,2,3,4,'sudh','Kumar',[1,2,3,4,5,6]]
```

NOTE How to write your own Datatype info in user define function.

```
ex. def test10(a,b):
      """ Function for addition of two
      Number """
      return a+b
```

Note :- there are two types of * static
① single * → store the tuple data
② double * → store the dict data type

scorser pointer → use shift & tabe button

M T W T F S S
Page No.:
Date:
YOUVA

test10(*)

Concept of *args :- pass any type of function in function

ex. # def test11(*args):
    return args
type(test11())
tuple
test11(1,2,3)
↳ (1,2,3)

test11(1,2,3,"Anil",[1,2,3,4],3.4)
↳ (1,2,3,'Anil',[1,2,3,4],3.4)

use **in Function

def test15(**kwargs):
    return kwargs
type(test15())
↳ dict
test15(a="sudh", [1,2,3,4] c=23.45)
↳ {'a':[1,2,3,4],'b':'sudh','c':23.45}

Lambda function :-
Lambda is also says Linear function, one Line function, anynomus function

ex. n=3
P=2
# def test(n,p):
    return n**p
test(3,2)
↳ 9

a = Lambda n,p : n**p
a(3,2)
↳ 9

add = Lambda x,y : x+y
add(4,5)
↳ 9

Statement :- Find tempature
c_to_f
ex. # c_to_f = Lambda c: (9/5)*c+32
c_to_f(45)
↳ 113

Finding_max = Lambda x,y : x if x>y else
Finding_max(34,23)
↳ 34

s = "pwskills"
Find_len = Lambda s : len(s)
Find_len(s)
↳ 8

Dictionory :-
→ Dictionory is a collection of keys values, used to store Data values Like a map, which, unlike other Data which hold only a single value as an element.
key should be Unqiue
ex. # d1 = {'key': "sudh"}
d1
↳ {'key': 'sudh'}

* Input function

marks = input("enter your marks")

Note. By Default input will get string value so convert str into integer

M T W T F S S | Page No.: | Date: | YOUVA

ex. * d2 = {'Name': "sudh", "email": "ss@gmail.com", "Number": 334534}

d2

↳ {'Name': 'sudh', 'email': ss@gmail.com', 'Number': 334534}

d2['Name']

↳ 'sudh'

d2['Number']

↳ 334534

key value respect with List

ds = {'company': 'pwskill', 'course': ['web dev', 'Data Science', 'Java with DSA']}

ds

↳ {'company': 'pwskill', 'course': ['web dev', 'Data science', 'Java with DSA']}

ds['course'][1]

↳ 'Data science'

d6 = {"Number": [2,34,3,34,34], 'assignment': (1,2,3,4,5,6), 'Launch-Date': {28,12,14}}

'classtime': {'web-dev': 8, 'Datascience': 10, 'Java with DSA': 12}}

d6['class-time']['data_science']

↳ 10

Add data in dictionary:

d6['mentor'] = ["sudh", "kirsh", 'anurag', 'hayder']}

↳ answer will add in d6.

Delete the key value pair in dictionary.

del d2['Name']

↳ {'email': ss@gmail.com', 'Number': 334534}

check the all key in d6

d6.keys()

↳ dict_keys(['Number', 'assignment', 'Launch_Date'])

d6.values() # d6.pop('key Name')

d6.items()

Control statement:

Control statements are used to control the Flow of the execution of the Loop based on a condition. there are many type of control statement.

① Break statement.

② Continue statement.

③ pass statement.

ex.

```
marks = 65
If marks >= 80:
    print("you will be a part of A0 batch")
elif marks >= 60 and marks < 80:
    print("you will be a part of A1 batch")
elif marks >= 40 and marks < 60:
    print("you will be a part of A2 batch")
else:
    print("you will be a part of A3 batch")
```

↳ 'you will be a part of A1 batch"}

range function genrate the Data using function

note. ex. range(0,5,1) ex. list(range(0,5,1))

start ↵ endpoint ↳ Jump → [0,1,2,3,4]

M T W T F S S

Page No.:

Date:

YOUVA

```
# price = int(input("enter price"))
# if price > 1000:
    print("I will not purchase")
else
    print("I will purchase")
# Enter price 500
↳ 'I will purchase'
```

```
# L = [1,2,3,4,5, "sudh", "kumar", 324.34,
    456, "abc"]
L1 = []
L2 = []
for i in l:
    if type(i) == int or type(i) == float:
        L1.append(i)
    else
        if type(i) == str:
            L2.append(i)
```

Loop:

Loop# are used to repeat a block of code. for example. If we want to show a message 100 times. then we can. use a Loop.

```
# L = [1,2,3,4,5,6,7,8]
L1 = []
for i in L:
    L1.append(i+1)
L1
↳ [2,3,4,5,6,7,8,9]
```

```
# L = ["sudh", "kumar", "pwskill",
"course"]
L1
for i in L:
    L1.append(i.upper())
L1
↳ ["SUDH", 'KUMAR', 'PWSKILL',
'COURSE']
```

for Loop:

```
L1 = ["sudh", "kumar", "kirsh", "Naik"]
for i in L1:
    print(i)
↳ 'sudh' 'kumar', 'Kirsh', 'Naik'
```

```
# use for else
for i in L1:
    if i == "kumaar":
        break
    print(i)
↳ 'sudh'
```

```
# use continue
for ir in L1:
    if i == "kumar":
        continue
    print(i)
↳ ' sudh . krish . NaiK
```

* reduce function always get two parameter first function & second is ~~Iterator~~.

M T W T F S S
Page No.:
Date:
YOUVA

range function :-

```
L1 = ['sudh', 'kumar', 'kirsh', 'Naik']
↳ =['sudh', 'kumar', 'kirsh', 'Naik']

for i in range(len(L1)-1,-1,-1):
    print(L1[i])
↳ Naik, kirsh, kumar, sudh.
```

Find sum of the List Using for Loop

```
# L = [1,2,3,4,7,8,98]
# sum(1)
↳ 129
# for i in L:
    result = result + i
  result
```

For Loop with String option.

```
For i in ~~in~~ S1 = "pwskills"
# For i in S1 :
    print(i)
↳ p w s k i l l s
```

for i in d.items()

```
('name','sudh')('class',' Data science master')
('topics',['python','ML','DL','CV',])
```

```
L = [2,3,4,5,6]
# map def sq(x):
    return x**2
# List(map(sq, l))
↳ [4, 9, 16, 25, 36]
```

Note :- map get the two parameter ① function ② Iterables.

```
# List(map(Lambda x**2, l))
↳ [4, 9, 16, 25, 36]
```

statement :- Find the add of two List element.

```
L1 = [1,2,3,4,5]
L2 = [6,7,8,9,10]
List(map(Lambda x,y : x+y, L1, L2))
↳ [7, 9, 11, 13, 15]
```

```
# def add(x,y):
    return x+y
# List(map(add, L1, L2))
↳ [7,9,11,13,15]
# s = 'pwskill"
# List(map(Lambda s : s.upper()))
↳ ['P','W','S','K','I','L','L']
```

Reduce :- reduce Function avaliable in from functools imports reduce

```
# L = [1,2,3,3,4,5]
# reduce(Lambda x,y : x+y, L)
↳ 18
# reduce(Lambda x,y : x if x > y
    else y, L)
↳ 5
# L
↳ [1,2,3,3,4,5]
```

Map, Reduce, Filter Function.

Map an Interator that computes the Function using arguments from each of Iterables. steps when the Shortest iterable is exhausted.

What is Fibonaic Number → Addition of previous two Number

Note ↳ Iteration : Find Data Using Next Next or by using iterFunction पे Data Interator के द्वारा Iterator Data Convert कर सकते हैं
Iterable :

Date:

* Filter Function :

```
# L = [1, 2, 3, 3, 4, 5]
# List (Filter x : x % 2 == 0 , L))
↳ [2, 4]
# List(filter x : x % 0 != 0, L)
↳ [3, 4, 5, 6, -1, -5]

# List( Filter(Lambda x : x < 0, L1))
↳ [-3, -1, -5]
# L2 = ["sudh", "pwskills", 'kumar', "bongalore"]
# List [ filter( Lambda x : len(x) < 6,
  L2]))
↳ [ 'sudh', 'kumar', 'kirsh']
```

* Generator Functions :-

- Generater function in python is a function that ~~reduce~~ returns an Iterator that produces a sequence of values when iterater over.

```
ex. #def test_fib (n):
        a, b = 0, 1
        for i in range (n):
            yield a
            a, b = b, a+b

# for i in test_Fib(10):
    print(i)
↳ [0 1 1 2 3 5. 8 13, 21, 34]
```

```
# Find fibonaic series using while loop.
def test_fib1():
    a, b = 0, 1
    while True :
        yield a
        a, b. = b, a+b
create the object of Fib = test-fib1()
```

```
for i in range (10):
    print(Next(fib))

# def count_test(n):
    count = 1
    while count <= n:
        yield count
        count = count +1
# c = count_test (5)
# type(c)
↳ generator
# For i in c:
    print (i)
1 2 3 4 5
```

"oops" stands for object oriented programming system.

class is Blueprint of class.

```
# class ~~pwskill~~ pwskill :
    def welcome_msg(self) :
        print(" welcome to pwskills")
# rohan = pwskill()
# rohan.welcome_msg()
↳ welcome to pwskills
# gurav = pwskill()
gurav. welcome_msg()
```

Init is a constructor.
self is not a keyword.

3 = 3x2x1
5 = 5x4x3x2x1
120 24 6

M T W T F S S

Page No.:
Date:
YOUVA

```
# class pwskills1 :
    def __init__(self, phone_number, email_id,
        student_id ):
        self. phone_number = phone_number
        self. email_id = email_id
        self. student_id = student_id
    def return_student_details(self):
        return self. student_id, self.phone_
    number , self. email_id
# rohan = pwskills1(1234567891, "rohan@
    gmail.com", 101)
# rohan.return_student_details()
↳ (101, 1234567891, "rohan@gmail.com")

# Ganesh = pwskills1( 2345678911, "Ganesh@
    gmail.com", 102)
# Ganesh.return_student_details()
↳ (102, 2345678911, "Ganesh@gmail.com")
```

while Loop:

~~while Loop is used to execute a block of statements repectedly until a~~ while Loop is used to run block of code until ~~to execute a block of statements~~ a certain condition is met.

```
er. a = 1
# while a<=10:
    print a<=10:
    print (a)
    a = a+1
↳ 1 2 3 4 5 6 7 8 9 10
```

```
# sum up the number till same point.
n = int (input ("enter your Limit"))
starting_point =0
counter =1
while counter <=n:
    starting_point = starting_point +
    counter.
    counter = counter+1
    starting_point
# Enter your Limit 5
↳ 15

# Factorical program you using while.
# Number = int( input ("enter your Number")
    factorial =1
    while number >0:
        factorial = factorial * Number
        number = number -1
    factorial.
Enter your Number 3
6

# Fibonacci series.
number = int ( input ("enter the Number
of element you are Looking for "))

Number = int ( input ("enter the Number
of element you are Looking for "))
a, b = 0,1
counter =0
while counter < Number :
    print (a)
```

my sql, postgress sql.
↳ Excel, python
rootcoss analysis, case study,
problem solving, softskills

4-3 0

Page No.:
Date:

C = a+b
a = b
b = c
counter = counter +1

enter the number of element you are
Looking for 5
0 ~~0~~ 1 2 3 5

reverse the string using while loop.

```
# word = input ("enter the string")
↳ enter the String "Anil"
# reverse = " "
  Length = Len (word)
  while Length > 0 :
      reverse = reverse + word [Length -1]
      Length = Length -1
  print (reverse)
↳ linA
```

A n i l
0 1 2 3

print table of Number

```
n = int (input ("enter your Number"))
i = 1
while i <= 10 :
    result = n * i
    print (n, "*", i, "=", result)
    i = i+1
```

Enter the Number 2
2 * 1 = 2
2 * 2 = 4
⋮
2 * 10 = 20

```
# n = 5
  i = 1
  while i < n :
      Print (i)
      i = i+1
  else :
      print ("while complete successfully")
```

↳ 1, 2, 3, 4
while complete successfully.

```
# n = 5
  i = 1
  while i < n
      Print (i)
      if i == 2 :
          break
  else :
      print ("this will be executed
      once you will complete it successfully")
```

break → thats why Loop will not completed!

↳ 1 2

polymorphism :
Different Different Behaviour in Different Different ~~situation~~ situation.

```
# def test (a, b) :
      return a+b
↳ 7
# test ("sudh", "kumar")
↳ 'sudh kumar'
```