```
# list, tuple, dict
# what is a list ?
# What type of values can be stored in list ?
# How do we create list - []
# List is mutable or immutable?
# How to check the length of the list ? - len()
# On list we can do indexing and slicing ? yes
# list_name[index]
# list_name[start_index : end_index : step]
# list functions -
# append -> adds one value at the end of the list
# extend -> adds more than one value at the end of the list
# insert -> inserts a value at a particular index (index, value)
# list_name[index] = new value
# remove -> remove(value)
# count -> count of value
# pop -> removes last value by default, but we can also specify the
index
# clear -> remove all the values from the list
# copy -> makes a copy of the list in different memory address
# sort -> sorts the list in ascending order by default,for descending
order we give reverse = True
#         sort cannot be used on mixed data type
# reverse -> reverses the list

# tuple ->
# What is tuple ?
# Tuple is a data structure in python , which once created cannot be
modified (immutable)
# We can do indexing and slicing on tuple
# tuple can store any kind of data
# length of the tuple - len()
# count - count of the value
# index - gives the index/position the the value

# min -> gives the min value from the tuple/list
# max -> gives the max value from the tuple/list
# sum -> gives the sum of the values in the tuple/list
# sorted -> gives the output in acending order by default, for
descending order specify reverse = True
#        always the output of sorted function is list

# dict
# A dictionary is a data structure in python which stores the value in
the form of key and value pair
# How to create a dict ?  - {}
# {'key' : 'value'}
# Is dict mutable or immutable? it is mutable
# What can be the key of the dictionary ?
# A key of the dictionary can be any immutable data type
```

```
# What can be the values of the dict?
# A value can be any data type
# Can we have duplicate keys in the dict? If there are duplicate keys
in dict it will take the last value
# {"A" : 10, "A" : 20}
# d['key'] = value, if key is already part of dict it will update the
value,
# if key is not part of the dict it will add new key and value in dict
# .keys()  -> it gives the key of the dict
# .vakues() -> it gives only the values of the dict
# .items() -> it gives both key and value in the form of tuple
# .get() -> gives the value of the specified key
# .pop('key') -> removes the given key and value from the dict
# .popitem()  -> by default removes the last key and value
# .update({}) -> updates the dict with the key and value
# .clear() -> removes all the key and value from the dict
# .copy() -> makes a copy in the memory
# dict.fromkeys()

d = {"A" : 10, "A" : 20}
d['A'] = 100
d
```

```
{'A': 100}
```

```
d['B'] = 200
d
```

```
{'A': 100, 'B': 200}
```

```
d.get("C", 'Key not found')
```

```
'Key not found'
```

```
d.pop('A')
```

```
100
```

```
d
```

```
{'B': 200}
```

```
my_list = ['Roy' ,'John', 'Steve']
dict.fromkeys(my_list)
```

```
{'Roy': None, 'John': None, 'Steve': None}
```

```
my_list = ('Roy' ,'John', 'Steve')
dict.fromkeys(my_list, 0)
```

```
{'Roy': 0, 'John': 0, 'Steve': 0}
```

```
s = 'python'
dict.fromkeys(s, 0)
```

```
{'p': 0, 'y': 0, 't': 0, 'h': 0, 'o': 0, 'n': 0}
```

```
### set

# set -
# A set is a data structure in python which can store only immutable
data types
# A set is unordered
set1 = {10, 20, 30}
set1
```

```
{10, 20, 30}
```

```
set2 = {10, 'python', 10.2, True}
set2
```

```
{10, 10.2, True, 'python'}
```

```
set2 = {10, 'python', 10.2, True, (10,30) }
set2
```

```
{(10, 30), 10, 10.2, True, 'python'}
```

```
set2 = {10, 'python', 10.2, True, [10,30] }
set2
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_3568/949027956.py in <module>
----> 1 set2 = {10, 'python', 10.2, True, [10,30] }
      2 set2

TypeError: unhashable type: 'list'
```

```
# can we store a set inside a set ?
# A set is mutable data type so a set cannot store a set
set3 = {10,20,{110,120} }
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_3568/3648734889.py in <module>
      1 # can we store a set inside a set ?
----> 2 set3 = {10,20,{110,120} }
```

```
TypeError: unhashable type: 'set'
```

```
set2
```

```
{(10, 30), 10, 10.2, True, 'python'}
```

```
len(set2)  # the length of set
```

```
5
```

```
# set functions -
# add, update, union, intersection, difference, issubset, issuperset
set_4 = {10,20,5}
set_4
```

```
{5, 10, 20}
```

```
# add - will add a value in the set
set_4.add(50)

print(set_4)
```

```
{10, 50, 20, 5}
```

```
# update -
# adds more than 1 value in set
set_4.update( {60, 70} )
print(set_4)
```

```
{50, 20, 5, 70, 10, 60}
```

```
# A set cannot store duplicate values
set_5 = {101, 102, 101}
set_5
```

```
{101, 102}
```

```
# union -
set_A = {10,20,40}
set_B = {101, 140, 80}

set_A.union(set_B)
```

```
{10, 20, 40, 80, 101, 140}
```

```
# union -
set_A = {10,20,40}
set_B = {10, 140, 80}

set_A.union(set_B)
```

```
{10, 20, 40, 80, 140}
```

```
# intersection
set_A.intersection(set_B)
```

```
{10}
```

```
# set difference
set_A.difference(set_B)
```

```
{20, 40}
```

```
# set difference
set_B.difference(set_A)
```

```
{80, 140}
```

```
# subset
set_C = {10,40,50,60,2,6}
set_D = {2,10}

set_C.issuperset(set_D)
```

```
True
```

```
set_E = {2, 10, 11}
set_E.issuperset(set_C)
```

```
False
```

```
set_D.issubset(set_C)
```

```
True
```

```
set_E.issubset(set_C)
```

```
False
```

```
# remove, discard, clear
set_C.remove(50)

set_C
```

```
{2, 6, 10, 40, 60}
```

```
set_C.discard(40)
print(set_C)
```

```
{2, 6, 10, 60}
```

```
set_C.clear()
print(set_C)
```

```
set()
```

```
set_E.remove(50)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_3568/1277677544.py in <module>
----> 1 set_E.remove(50)

KeyError: 50
```

```
set_E.discard(50)
```

```
set_E
```

```
{2, 10, 11}
```

```
# remove will throw an error if we try to remove a value which is not part of the set
# discard will not throw any error even if we try to remove a value which is not part of the set
```

```
min(set_E)
```

```
2
```

```
max(set_E)
```

```
11
```

```
sum(set_E)
```

```
23
```

```
sorted(set_E)
```

```
[2, 10, 11]
```

```
# converting a list into tuple and set and dict, str
my_list = ['Roy', 'John', 'Steve']
my_list
```

```
['Roy', 'John', 'Steve']
```

```
type(my_list)
```

```
list
```

```
tuple(my_list)
```

```
('Roy', 'John', 'Steve')
```

```
set(my_list)
```

```
{'John', 'Roy', 'Steve'}
```

```
str(my_list)
```

```
"['Roy', 'John', 'Steve']"
```

```
dict.fromkeys(my_list)
```

```
{'Roy': None, 'John': None, 'Steve': None}
```

```
# a tuple can be converted into list, set, str, dict
tuple_1 = (10,20,60)
tuple_1
```

```
(10, 20, 60)
```

```
list(tuple_1)
```

```
[10, 20, 60]
```

```
set(tuple_1)
```

```
{10, 20, 60}
```

```
str(tuple_1)
```

```
'(10, 20, 60)'
```

```
dict.fromkeys(tuple_1)
```

```
{10: None, 20: None, 60: None}
```

```
# set can be converted to list, tuple, str, dict
```

```
set_A
```

```
{10, 20, 40}
```

```
list(set_A)
```

```
[40, 10, 20]
```

```
tuple(set_A)
```

```
(40, 10, 20)
```

```
str(set_A)
```

```
'{40, 10, 20}'
```

```
dict.fromkeys(set_A)
```

```
{40: None, 10: None, 20: None}
```

```
# Can we convert a dict into list, tuple, set, str
d = {'A' : 20, 'B' : 30, 'C' : 40}
d
```

```
{'A': 20, 'B': 30, 'C': 40}
```

```
list(d)
```

```
['A', 'B', 'C']
```

```
tuple(d)
```

```
('A', 'B', 'C')
```

```
set(d)
```

```
{'A', 'B', 'C'}
```

```
str(d)
```

```
"{'A': 20, 'B': 30, 'C': 40}"
```

```
# can we convert a string into list, tuple, set, dict
s
```

```
'python'
```

```
list(s)
```

```
['p', 'y', 't', 'h', 'o', 'n']
```

```
tuple(s)
```

```
('p', 'y', 't', 'h', 'o', 'n')
```

```
set(s)
```

```
{'h', 'n', 'o', 'p', 't', 'y'}
```

```
dict.fromkeys(s)
```

```
{'p': None, 'y': None, 't': None, 'h': None, 'o': None, 'n': None}
```

```
# membership operators on sequence/iterbale data types (string, list,
tuple, set, dict)

# in, not in
s
```

```
'python'
```

```
'p' in s
```

```
True
```

```
'py' in s
```

```
True
```

```
'po' in s
```

```
False
```

```
'po' not in s
```

```
True
```

```
my_list
```

```
['Roy', 'John', 'Steve']
```

```
'Roy' in my_list
```

```
True
```

```
'max' in my_list
```

```
False
```

```
'max' not in my_list
```

```
True
```

```
tuple_1
```

```
(10, 20, 60)
```

```
10 in tuple_1
```

```
True
```

```
set_A
```

```
{10, 20, 40}
```

```
10 in set_A
```

```
True
```

```
d
```

```
{'A': 20, 'B': 30, 'C': 40}
```

```
'A' in d
```

```
True
```

```
20 in d
```

```
False
```

```
# create a string - 'python for data science'
# find the count of unique characters in the string
# convert each word into a list of word
# create a dictonary with key as the words from the dict and values as
none.

string = 'python for data science'

# output 2 - ['python', 'for', 'data', 'science']
# output 3 - {'python' : None, 'for' : None, 'data' : None,
'science' :  None}

# create a string - 'python for data science'
string = 'python for data science'
string
```

```
'python for data science'
```

```
# find the count of unique characters in the string
# A set cannot store duplicate values
a = set(string)
a
```

```
{' ', 'a', 'c', 'd', 'e', 'f', 'h', 'i', 'n', 'o', 'p', 'r', 's', 't',
'y'}
```

```
len(a)
```

```
15
```

```
# convert each word into a list of word
string
```

```
'python for data science'
```

```
b = string.split()
b
```

```
['python', 'for', 'data', 'science']
```

```
dict.fromkeys(b)
```

```
{'python': None, 'for': None, 'data': None, 'science': None}
```

```
# string1 = 'Hello, python'
# string2 = 'Hello, java'
# find the unique charcters in string1 and string2
# find the unique charcters in both string1 and string2
# find the common characters in string1 and string2
```

```
string1 = 'Hello, python'
string2 = 'Hello, java'

a = set(string1)
a
```

```
{' ', ',', 'H', 'e', 'h', 'l', 'n', 'o', 'p', 't', 'y'}
```

```
b = set(string2)
b
```

```
{' ', ',', 'H', 'a', 'e', 'j', 'l', 'o', 'v'}
```

```
a.union(b)
```

```
{' ', ',', 'H', 'a', 'e', 'h', 'j', 'l', 'n', 'o', 'p', 't', 'v', 'y'}
```

```
s = string1+string2
set(s)
```

```
{' ', ',', 'H', 'a', 'e', 'h', 'j', 'l', 'n', 'o', 'p', 't', 'v', 'y'}
```

```
a
```

```
{' ', ',', 'H', 'e', 'h', 'l', 'n', 'o', 'p', 't', 'y'}
```

```
b
```

```
{' ', ',', 'H', 'a', 'e', 'j', 'l', 'o', 'v'}
```

```
a.intersection(b)
```

```
{' ', ',', 'H', 'e', 'l', 'o'}
```

```
### Conditional Statements ####

# if condition to check :
#     program
# else :
#    program

# If you score 18 and above you have passed your exam otherwise fail

marks = 20

if marks >= 18:
    print("Pass")
else:
    print("Fail")

print("Done checking!")
```

```
Pass
Done checking!
```

```
marks = 15

if marks >= 18:
    print("Pass")
else:
    print("Fail")

print("Done checking!")
```

```
Fail
Done checking!
```

```
# Write a program to check if a person is eligible to vote or not
# Check if a number is odd or even

age = 15

if age>=18:
    print('Eligible')
else:
    print('Not Eligible')
```

```
Not Eligible
```

```


# Check if a number is odd or even
number = 12
if number % 2 == 0:
    print('Even')
else:

    print('Odd')
```

```
Even
```

```
# if number is even add 10 to it and print it, if number is odd add 5
to it and print it

number = 13

if number % 2 == 0:
    number = number + 10
    print(number)
else:
    number = number + 5
    print(number)
```

```
18
```

```python
# If a number is even, add 10  to that number and print the number
# if it is odd, add 5 to that number and print the number
number = 13
if number % 2 == 0:
    number = number + 10
    print(number)
else:
    number = number + 5
    print(number)
```

```
18
```

```python
# We can also have multiple conditions to check
age = 18
country = 'australia'

if (age >= 18)  and (country == 'india') :
    print("Eligible")
else:
    print('Not Eligible')
```

```
Not Eligible
```

```python
# Check if a number is even and grater than 10, if yes print even and
greater than 10
# if not print odd or less than 10

# check if python is part for the string s, if it is part of string
print python is present in
# string, else print python is not present in string

s = 'Data Analytics using R programming'

if 'python' in s:
    print("python is present in string")
else:
    print('python is not present in string')
```

```
python is not present in string
```

```python


# nested if else conditions

marks = int(input("Enter your marks - "))

if marks >= 33:
    print('Pass')
else:
    print("Fail")
```

```
Enter your marks - 150
Pass
```

```
marks = int(input("Enter your marks - "))

if (marks >= 0) and (marks <=100):

    print('Valid Input!')

    if marks >= 33:
        print("Pass")
    else:
        print('Fail')

else:
    print('Invalid Input')
```

```
Enter your marks - 23
Valid Input!
Fail
```

```
# check if age is valid (between 0 to 120), if age is valid are they
eligible to vote

# if, elif, else (nested conditional statements)

marks = int(input("Enter your marks - "))

if (marks<0)  or (marks > 100):
    print('Invalid Input')
elif marks >= 33:
    print('Pass')
else:
    print('Fail')
```

```
Enter your marks - 32
Fail
```

```
age = int(input("Enter your age - "))
if (age<0) or (age>120):
    print('Invalid Input')
elif age >= 18:
    print('Eligible')
else:
    print('Not eligible')
```

```
Enter your age - 12
Not eligible
```