Basic data structure in Python

- 1. Tuple
- 1. List
- 1. Dictionaries

TUPLE :-

- ordered collection of elements
- enclosed in () rounds braces / paranthesis
- Different kind of elements can be stored
- Once elements are stored you cannot change them (immutable)

In [1]:
```
tup1 = (1 , "python" , True , 2.5)
tup1
```

Out[1]: (1, 'python', True, 2.5)

In [2]:
```
# Type of tuple :-
type (tup1)
```

Out[2]: tuple

- Indexing in tuple

In [3]:
```
tup1[0]
```

Out[3]: 1

In [4]:
```
tup1[2]
```

Out[4]: True

In [5]:
```
tup1[0:4]
```

Out[5]: (1, 'python', True, 2.5)

In [6]:
```
# last element is exclusive
tup1[0:3]
```

Out[6]: (1, 'python', True)

In [7]:
```
# count of elements in tuple
len(tup1)
```

Out[7]: 4

In [8]:
```
tup2 =(2 , "babaAmmar" , 3.5 , False )
```

In [9]:
```
# concatenate or addition or merging
tup1 + tup2
```

Out[9]: (1, 'python', True, 2.5, 2, 'babaAmmar', 3.5, False)

In [10]:
```
tup1*2 + tup2
```

Out[10]: (1, 'python', True, 2.5, 1, 'python', True, 2.5, 2, 'babaAmmar', 3.5, False)

In [13]:
```
tup3 = (20,60,30,60,79,85)
tup3
```

Out[13]: (20, 60, 30, 60, 79, 85)

In [14]:
```
# Minimum value
min(tup3)
```

Out[14]: 20

In [15]:
```
# Max value
max(tup3)
```

Out[15]: 85

In [16]:
```
tup3*2
```

Out[16]: (20, 60, 30, 60, 79, 85, 20, 60, 30, 60, 79, 85)

In [68]:
```
# The count() method returns the number of times the specified element appear
tup3.count(20)
```

Out[68]: 1

In [70]:
```
# The index() method returns the index of the specified element in the tuple.
# vowels tuple
vowels = ('a', 'e', 'i', 'o', 'i', 'u')

# index of 'e' in vowels
index = vowels.index('e')
print('The index of e:', index)

# element 'i' is searched
# index of the first 'i' is returned
index = vowels.index('i')

print('The index of i:', index)
```

```
The index of e: 1
The index of i: 2
```

LISTS

- ordered collection of elements
- enclosed in [] Square brackets
- mutable , you can change values

In [17]:
```
list1 = [2 , "babaAmmar" , False]
list1
```

Out[17]: [2, 'babaAmmar', False]

In [18]:
```
type(list1) # check type
```

Out[18]: list

In [19]:
```
len(list1) # check length
```

Out[19]: 3

In [20]:
```
list1[2]
```

Out[20]: False

In [22]:
```
list2 = [3, 6 , "ammar" , "codanics", 475 , 53.2 , False]
list2
```

Out[22]: [3, 6, 'ammar', 'codanics', 475, 53.2, False]

In [23]:
```
list1 + list2
```

Out[23]:
```
[2, 'babaAmmar', False, 3, 6, 'ammar', 'codanics', 475, 53.2, False]
```

In [24]:
```
# list ko 2 se multiply karny se 2 dafa ajaiga sab
list1*2
```

Out[24]:
```
[2, 'babaAmmar', False, 2, 'babaAmmar', False]
```

In [27]:
```
list1
```

Out[27]:
```
[2, 'babaAmmar', False]
```

In [26]:
```
list1.reverse()
list1
```

Out[26]:
```
[2, 'babaAmmar', False]
```

In [28]:
```
list1.append( "codanics youtube channel")
list1
```

Out[28]:
```
[2, 'babaAmmar', False, 'codanics youtube channel']
```

In [34]:
```
list1.count(2)
#The count() method returns the number of times the specified element appears
```

Out[34]:
```
1
```

In [35]:
```
#The extend() method adds all the elements of an iterable (list, tuple, strin
# create a list
prime_numbers = [2, 3, 5]

# create another list
numbers = [1, 4]

# add all elements of prime_numbers to numbers
numbers.extend(prime_numbers)


print('List after extend():', numbers)

# Output: List after extend(): [1, 4, 2, 3, 5]
```

```
List after extend(): [1, 4, 2, 3, 5]
```

In [36]:
```
# The index() method returns the index of the specified element in the list.
animals = ['cat', 'dog', 'rabbit', 'horse']

# get the index of 'dog'
index = animals.index('dog')


print(index)

# Output: 1
```

```
1
```

In [37]:
```
# The insert() method inserts an element to the list at the specified index.
# create a list of vowels
vowel = ['a', 'e', 'i', 'u']

# 'o' is inserted at index 3 (4th position)
vowel.insert(3, 'o')


print('List:', vowel)

# Output: List: ['a', 'e', 'i', 'o', 'u']
```

```
List: ['a', 'e', 'i', 'o', 'u']
```

In [38]:
```
# The pop() method removes the item at the given index from the list and retu
# create a list of prime numbers
prime_numbers = [2, 3, 5, 7]

# remove the element at index 2
removed_element = prime_numbers.pop(2)

print('Removed Element:', removed_element)
print('Updated List:', prime_numbers)

# Output:
# Removed Element: 5
# Updated List: [2, 3, 7]
```

```
Removed Element: 5
Updated List: [2, 3, 7]
```

In [39]:
```
# The remove() method removes the first matching element (which is passed as 
# create a list
prime_numbers = [2, 3, 5, 7, 9, 11]

# remove 9 from the list
prime_numbers.remove(9)


# Updated prime_numbers List
print('Updated List: ', prime_numbers)

# Output: Updated List:  [2, 3, 5, 7, 11]
```

```
Updated List:  [2, 3, 5, 7, 11]
```

In [42]:
```
list3 = [20 , 30 , 35 , 50 , 40 , 12 , 15 , 31 , 10 , 356 , 886]
list3
l3 = list3
l3
```

Out[42]:
```
[20, 30, 35, 50, 40, 12, 15, 31, 10, 356, 886]
```

In [43]:
```
# Sorting a list , it will sort in ascending order
l3.sort()
l3
```

Out[43]:
```
[10, 12, 15, 20, 30, 31, 35, 40, 50, 356, 886]
```

In [44]:
```
l3*2
```

Out[44]:
```
[10,
 12,
 15,
 20,
 30,
 31,
 35,
 40,
 50,
 356,
 886,
 10,
 12,
 15,
 20,
 30,
 31,
 35,
 40,
 50,
 356,
 886]
```

In [2]:
```
# Python List sort()
# The sort() method sorts the elements of a given list in a specific ascendin

prime_numbers = [11, 3, 7, 5, 2]

# sort the list
prime_numbers.sort()

print(prime_numbers)

# Output: [2, 3, 5, 7, 11]
```

```
[2, 3, 5, 7, 11]
```

Dictionaries

- un ordered collection of elements
- key and values
- curely brackets {}{}
- mutable , change values

In [50]:
```
# Food and their prices
food1 = {"samosa":10 , "pakora":100 , "raita":20 , "salad":50 , "pakora":100}
food1
```

Out[50]:
```
{'samosa': 10, 'pakora': 100, 'raita': 20, 'salad': 50}
```

In [51]:
```
type(food1)
```

Out[51]:
```
dict
```

In [53]:
```
# data extract b kar skty
keys1 = food1.keys()
keys1
```

Out[53]:
```
dict_keys(['samosa', 'pakora', 'raita', 'salad'])
```

In [55]:
```
values1 = food1.values()
values1
```

Out[55]:
```
dict_values([10, 100, 20, 50])
```

In [ ]:
```
food1.update
```

In [57]:
```
# we can mutatte or add new
food1["tikki"]=10
food1
```

Out[57]:
```
{'samosa': 10, 'pakora': 100, 'raita': 20, 'salad': 50, 'tikki': 10}
```

In [58]:
```
# update the values
food1["tikki"] = 15
food1
```

Out[58]:
```
{'samosa': 10, 'pakora': 100, 'raita': 20, 'salad': 50, 'tikki': 15}
```

In [59]:
```
food2 = {"dates":50 , "choclates":200 , "sawayn":1000}
food2
```

Out[59]:
```
{'dates': 50, 'choclates': 200, 'sawayn': 1000}
```

In [61]:
```
# concatenate , dictionaries jo jama karny k lye following update karna pareg
food1.update(food2)
food1
```

Out[61]:
```
{'samosa': 10,
 'pakora': 100,
 'raita': 20,
 'salad': 50,
 'tikki': 15,
 'dates': 50,
 'choclates': 200,
 'sawayn': 1000}
```

In [3]:
```
# Python Dictionary values()
# The values() method returns a view object that displays a list of all the v
marks = {'Physics':67, 'Maths':87}

print(marks.values())


# Output: dict_values([67, 87])
```

```
dict_values([67, 87])
```

In [4]:
```
# random sales dictionary
sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }

print(sales.values())
```

```
dict_values([2, 3, 4])
```

In [5]:
```
# Python Dictionary clear()

# The clear() method removes all items from the dictionary.
d = {1: "one", 2: "two"}

d.clear()
print('d =', d)
```

```
d = {}
```

In [7]:
```
# Python Dictionary copy()

# They copy() method returns a copy (shallow copy) of the dictionary.

original = {1:'one', 2:'two'}
new = original.copy()


print('Orignal: ', original)
print('New: ', new)
```

```
Orignal:  {1: 'one', 2: 'two'}
New:  {1: 'one', 2: 'two'}
```

In [8]:
```
# Using = Operator to Copy Dictionaries

original = {1:'one', 2:'two'}
new = original


# removing all elements from the list
new.clear()

print('new: ', new)
print('original: ', original)
```

```
new:  {}
original:  {}
```

In [9]:
```
# Using copy() to Copy Dictionaries

original = {1:'one', 2:'two'}
new = original.copy()


# removing all elements from the list
new.clear()

print('new: ', new)
print('original: ', original)
```

```
new:  {}
original:  {1: 'one', 2: 'two'}
```

In [11]:
```
# Python Dictionary fromkeys()

# The fromkeys() method creates a new dictionary from the given sequence of e

# vowels keys
keys = {'a', 'e', 'i', 'o', 'u' }

vowels = dict.fromkeys(keys)
print(vowels)
```

```
{'o': None, 'e': None, 'i': None, 'a': None, 'u': None}
```

In [12]:
```
# Create a dictionary from a sequence of keys with value
#  vowels keys
keys = {'a', 'e', 'i', 'o', 'u' }
value = 'vowel'

vowels = dict.fromkeys(keys, value)
print(vowels)
```

```
{'o': 'vowel', 'e': 'vowel', 'i': 'vowel', 'a': 'vowel', 'u': 'vowel'}
```

In [13]:
```
 # Create a dictionary from mutable object list

 # vowels keys
keys = {'a', 'e', 'i', 'o', 'u' }
value = [1]

vowels = dict.fromkeys(keys, value)
print(vowels)

# updating the value
value.append(2)
print(vowels)
```

```
{'o': [1], 'e': [1], 'i': [1], 'a': [1], 'u': [1]}
{'o': [1, 2], 'e': [1, 2], 'i': [1, 2], 'a': [1, 2], 'u': [1, 2]}
```

In [15]:
```
# Python Dictionary items()
# The items() method returns a view object that displays a list of dictionary

#  Get all items of a dictionary with items()

# random sales dictionary
sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }

print(sales.items())
```

```
dict_items([('apple', 2), ('orange', 3), ('grapes', 4)])
```

In [19]:
```
# Python Dictionary popitem()

# The Python popitem() method removes and returns the last element (key, value

person = {'name': 'Phill', 'age': 22, 'salary': 3500.0}

# ('salary', 3500.0) is inserted at the last, so it is removed.
result = person.popitem()

print('Return Value = ', result)
print('person = ', person)

# inserting a new element pair
person['profession'] = 'Plumber'

# now ('profession', 'Plumber') is the latest element
result = person.popitem()

print('Return Value = ', result)
print('person = ', person)
```

```
Return Value =  ('salary', 3500.0)
person =  {'name': 'Phill', 'age': 22}
Return Value =  ('profession', 'Plumber')
person =  {'name': 'Phill', 'age': 22}
```

In [20]:
```
# Python Dictionary setdefault()

# The setdefault() method returns the value of a key (if the key is in diction

# How setdefault() works when key is in the dictionary?

person = {'name': 'Phill', 'age': 22}

age = person.setdefault('age')
print('person = ',person)
print('Age = ',age)
```

```
person =  {'name': 'Phill', 'age': 22}
Age =  22
```

In [22]:
```
#  How setdefault() works when key is not in the dictionary?

person = {'name': 'Phill'}

# key is not in the dictionary
salary = person.setdefault('salary')
print('person = ',person)
print('salary = ',salary)

# key is not in the dictionary
# default_value is provided
age = person.setdefault('age', 22)
print('person = ',person)
print('age = ',age)
```

```
person =  {'name': 'Phill', 'salary': None}
```

```
salary =  None
person =  {'name': 'Phill', 'salary': None, 'age': 22}
age =  22
```

In [23]:

```
# Python Dictionary update()
# The update() method updates the dictionary with the elements from another d

marks = {'Physics':67, 'Maths':87}
internal_marks = {'Practical':48}

marks.update(internal_marks)


print(marks)

# Output: {'Physics': 67, 'Maths': 87, 'Practical': 48}
```

```
{'Physics': 67, 'Maths': 87, 'Practical': 48}
```

In [24]:

```
d = {1: "one", 2: "three"}
d1 = {2: "two"}

# updates the value of key 2
d.update(d1)

print(d)

d1 = {3: "three"}

# adds element with key 3
d.update(d1)

print(d)
```

```
{1: 'one', 2: 'two'}
{1: 'one', 2: 'two', 3: 'three'}
```

SETS

- un ordred and un index set of elements
- curly braces are used {}
- no duplicates allowed
- keys or values wala scene yaha nai hy

In [64]:

```
s1 = {3 , 2.2 , 5.2 , "Aammar", "codanics" , "peshawar"}
s1
```

Out[64]:

```
{2.2, 3, 5.2, 'Aammar', 'codanics', 'peshawar'}
```

In [65]:

```
# how to add something
s1.add("asad")
s1
```

Out[65]:
```
{2.2, 3, 5.2, 'Aammar', 'asad', 'codanics', 'peshawar'}
```

In [66]:
```
# how to remove
s1.remove("Aammar")
s1
```

Out[66]:
```
{2.2, 3, 5.2, 'asad', 'codanics', 'peshawar'}
```

In [71]:
```
#Python Set update()

#The Python set update() method updates the set, adding items from other itera

A = {'a', 'b'}
B = {1, 2, 3}

result = A.update(B)

print('A =', A)
print('result =', result)
```

```
A = {1, 2, 3, 'a', 'b'}
result = None
```

In [72]:
```
string_alphabet = 'abc'
numbers_set = {1, 2}

# add elements of the string to the set
numbers_set.update(string_alphabet)

print('numbers_set =', numbers_set)

info_dictionary = {'key': 1, 'lock' : 2}
numbers_set = {'a', 'b'}

# add keys of dictionary to the set
numbers_set.update(info_dictionary)
print('numbers_set =', numbers_set)
```

```
numbers_set = {1, 2, 'c', 'a', 'b'}
numbers_set = {'b', 'key', 'lock', 'a'}
```

In [73]:
```
# Python frozenset()

# The frozenset() function returns an immutable frozenset object initialized w
# tuple of vowels
vowels = ('a', 'e', 'i', 'o', 'u')

fSet = frozenset(vowels)
print('The frozen set is:', fSet)
print('The empty frozen set is:', frozenset())

# frozensets are immutable
fSet.add('v')
```

```
The frozen set is: frozenset({'e', 'u', 'i', 'a', 'o'})
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-73-dae7dfb15af8> in <module>
     10
     11 # frozensets are immutable
---> 12 fSet.add('v')

AttributeError: 'frozenset' object has no attribute 'add'
```

In [74]:

```
# random dictionary
person = {"name": "John", "age": 23, "sex": "male"}

fSet = frozenset(person)
print('The frozen set is:', fSet)
```

```
The frozen set is: frozenset({'age', 'sex', 'name'})
```

In [75]:

```
# Python Set add()
# The add() method adds a given element to a set. If the element is already p

prime_numbers = {2, 3, 5, 7}

# add 11 to prime_numbers
prime_numbers.add(11)


print(prime_numbers)

# Output: {2, 3, 5, 7, 11}
```

```
{2, 3, 5, 7, 11}
```

In [76]:

```
# Python Set clear()

# The clear() method removes all elements from the set.

# set of vowels
vowels = {'a', 'e', 'i', 'o', 'u'}
print('Vowels (before clear):', vowels)

# clearing vowels
vowels.clear()
print('Vowels (after clear):', vowels)
```

```
Vowels (before clear): {'e', 'u', 'i', 'a', 'o'}
Vowels (after clear): set()
```

In [78]:
```
# Python Set copy()

# The copy() method returns a shallow copy of the set.
numbers = {1, 2, 3, 4}
new_numbers = numbers

new_numbers.add(5)

print('numbers: ', numbers)
print('new_numbers: ', new_numbers)
new_numbers.copy()
```

```
numbers:  {1, 2, 3, 4, 5}
new_numbers:  {1, 2, 3, 4, 5}
```

Out[78]:
```
{1, 2, 3, 4, 5}
```

In [81]:
```
# Python Set difference()

# The difference() method returns the set difference of two sets.

A = {'a', 'b', 'c', 'd'}
B = {'c', 'f', 'g'}

# Equivalent to A-B
print(A.difference(B))

# Equivalent to B-A
print(B.difference(A))
```

```
{'b', 'a', 'd'}
{'f', 'g'}
```

In [82]:
```
# Python Set discard()

# The discard() method removes a specified element from the set (if present).

numbers = {2, 3, 5, 4}

# Returns None
# Meaning, absence of a return value
print(numbers.discard(3))

print('numbers =', numbers)
```

```
None
numbers = {2, 4, 5}
```

In [83]:

```
# Python Set intersection()
# The intersection() method returns a new set with elements that are common t

A = {2, 3, 5, 4}
B = {2, 5, 100}
C = {2, 3, 8, 9, 10}

print(B.intersection(A))
print(B.intersection(C))
print(A.intersection(C))

print(C.intersection(A, B))
```

```
{2, 5}
{2}
{2, 3}
{2}
```

In [84]:

```
# Python Set isdisjoint()

# The isdisjoint() method returns True if two sets are disjoint sets. If not,

A = {'a', 'b', 'c', 'd'}
B = ['b', 'e', 'f']
C = '5de4'
D ={1 : 'a', 2 : 'b'}
E ={'a' : 1, 'b' : 2}

print('Are A and B disjoint?', A.isdisjoint(B))
print('Are A and C disjoint?', A.isdisjoint(C))
print('Are A and D disjoint?', A.isdisjoint(D))
print('Are A and E disjoint?', A.isdisjoint(E))
```

```
Are A and B disjoint? False
Are A and C disjoint? False
Are A and D disjoint? True
Are A and E disjoint? False
```

In [85]:
```
# Python Set issubset()

# The issubset() method returns True if all elements of a set are present in 

A = {1, 2, 3}
B = {1, 2, 3, 4, 5}
C = {1, 2, 4, 5}

# Returns True
print(A.issubset(B))

# Returns False
# B is not subset of A
print(B.issubset(A))

# Returns False
print(A.issubset(C))

# Returns True
print(C.issubset(B))
```

```
True
False
False
True
```

In [86]:
```
# Python Set issuperset()

# The issuperset() method returns True if a set has every elements of another

A = {1, 2, 3, 4, 5}
B = {1, 2, 3}
C = {1, 2, 3}

# Returns True
print(A.issuperset(B))

# Returns False
print(B.issuperset(A))

# Returns True
print(C.issuperset(B))
```

```
True
False
True
```

In [87]:
```
# Python Set pop()

# The pop() method removes an arbitrary element from the set and returns the 


A ={'a', 'b', 'c', 'd'}

print('Return Value is', A.pop())
print('A = ', A)
```

```
Return Value is b
A =  {'c', 'a', 'd'}
```

In [91]:

```
# Python Set remove()
# The remove() method removes the specified element from the set.


# language set
language = {'English', 'French', 'German'}

# removing 'German' from language
language.remove('German')


# Updated language set
print('Updated language set:', language)
```

```
Updated language set: {'English', 'French'}
```

In [94]:

```
# Python Set union()
# The Python set union() method returns a new set with distinct elements from

A = {'a', 'c', 'd'}
B = {'c', 'd', 2 }
C = {1, 2, 3}

print('A U B =', A.union(B))
print('B U C =', B.union(C))
print('A U B U C =', A.union(B, C))

print('A.union() =', A.union())
```

```
A U B = {'d', 2, 'a', 'c'}
B U C = {'d', 1, 2, 3, 'c'}
A U B U C = {'d', 1, 2, 3, 'a', 'c'}
A.union() = {'d', 'c', 'a'}
```

In [95]:

```
# Python Set symmetric_difference()

# The Python symmetric_difference() method returns the symmetric difference o

A = {'a', 'b', 'c', 'd'}
B = {'c', 'd', 'e' }
C = {}

print(A.symmetric_difference(B))
print(B.symmetric_difference(A))

print(A.symmetric_difference(C))
print(B.symmetric_difference(C))
```

```
{'a', 'b', 'e'}
{'a', 'b', 'e'}
{'b', 'a', 'c', 'd'}
{'d', 'c', 'e'}
```

In [ ]: