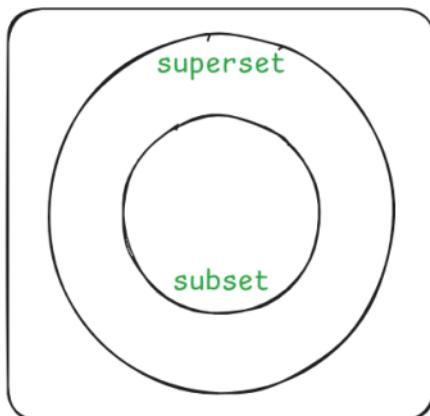
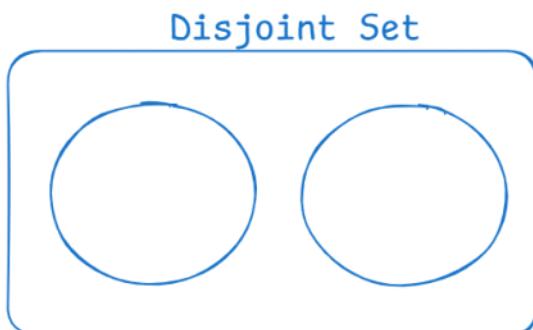
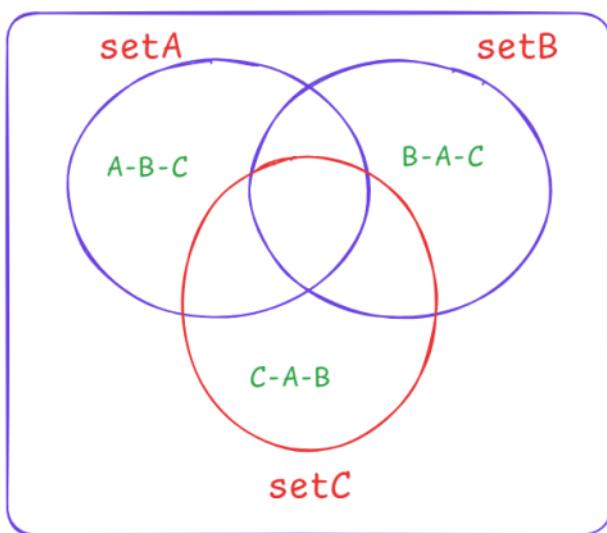


Sets and Dictionary in Python

Session Objectives

- █ Understand what sets are.
- █ Understand common methods and operations associated with sets.
- █ Understand the comparison between lists, tuples and sets.
- █ Understand what dictionaries are.
- █ Understand common methods and operations associated with dictionaries.
- █ Understand the comparison between lists, tuples, sets and dictionaries.



```
setA = setA ^ (setB ^ setC)
(setB ^ setC) = (setB-setC) U (setC-setB)
setA = setA ^ (setB-setC) U (setC-setB)
```

```

# difference_update()
# shortcut : a -= b/c => a = a - (b/c)
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
# setB | setC = {a,b,c,p,q,r,s,x,y,z}
# setA - (setB | setC) = {'a', 'b', 'c', 'd', 'e'} - {a,b,c,p,q,r,s,x,y,z}
# setA = {'d', 'e'}
setA = setA - (setB | setC)
print(setA) # {d,e}

{'d', 'e'}

setA.difference_update(setB, setC)
print(setA) # {d,e}

{'d', 'e'}

```

```

setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
# setA | setC = {a,b,c,d,e,r,s,x,y,z}
# setB - (setA | setC) = {'p', 'q', 'r', 's', 'a', 'b', 'c'} - {a,b,c,d,e,r,s,x,y,z}
# setB = {p,q}
setB = setB - (setA | setC)
print(setB) # {p,q}

{'p', 'q'}
```

intersection_update()

```

# shortcut : a &= b => a = a & b
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
setA.intersection_update(setB) # {'a', 'b', 'c'}
print(setA)

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

```

setA = setA & setB
print(setA)

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

intersection_update()

```

# shortcut : a&= b&c => a = a & (b&c)
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
setA.intersection_update(setB, setC) # {'a', 'c'}
# {'a', 'b', 'c', 'd', 'e'} & ({'a', 'c', 'r', 's'}) # {'a', 'c'}
print(setA)

{'c', 'a'}
```

```

setA = setA & setB & setC
print(setA)

{'c', 'a'}
```

```
# isdisjoint() -> Boolean Return [True / False]
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
setD = {'k', 'm'}
print(setA.isdisjoint(setB)) # False
print(setA.isdisjoint(setC)) # False
print(setB.isdisjoint(setC)) # False
print(setB.isdisjoint(setD)) # True
print(setA.isdisjoint(setD)) # True
```

```
False
False
False
True
True
```

```
# issubset() & issuperset()
set1 = {1,2,3,4,5}
set2 = {1,2,3,4,5,6,7,8,9}
print(set1.issubset(set2)) # True as set1 is a subset of set2
print(set2.issuperset(set1)) # True as set2 is a superset of set1
```

```
True
True
```

```
print(set2.issubset(set1)) # False
print(set1.issuperset(set2)) # False
```

```
False
False
```

```
# symmetric_difference_update() : '^'
# shortcut : a ^= b => a = a^b
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
setA.symmetric_difference_update(setB) # {d,e} U {p,q,r,s}
print(setA) # {depqrs}
```

```
{'r', 'p', 'q', 's', 'd', 'e'}
```

```
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
setA = setA ^ setB
print(setA)
```

```
{'r', 'p', 'q', 's', 'd', 'e'}
```

```
# symmetric_difference_update() : '^'
```

```

# symmetric_difference_update() : '^'
# shortcut : a ^= (b^c) => a = a^(b^c)
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
# setB ^ setC = {pqb,xyz}
# setA = setA ^ {pqb,xyz} # {'a', 'b', 'c', 'd', 'e'} ^ {pqb,xyz}
# setA = {a,c,d,e,pq,xyz}
setA = setA ^ (setB ^ setC) # {acde,pq,xyz}
print(setA)

{'q', 'a', 'x', 'c', 'p', 'd', 'e', 'z', 'y'}

setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
print(setA.symmetric_difference(setB).symmetric_difference(setC))

{'q', 'a', 'x', 'c', 'p', 'd', 'e', 'z', 'y'}

```

```

setA = setA ^ (setB ^ setC)
(setB ^ setC) = (setB-setC) U (setC-setB)
setA = setA ^ (setB-setC) U (setC-setB)

# update() '/ '
# shortcut : a/=(b/c) => a = a / (b/c)
setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
setA.update(setB, setC)
print(setA) # {abcde,pqrs,xyz}

{'r', 'p', 'a', 'q', 's', 'd', 'e', 'x', 'b', 'z', 'c', 'y'}

setA = {'a', 'b', 'c', 'd', 'e'}
setB = {'p', 'q', 'r', 's', 'a', 'b', 'c'}
setC = {'x', 'y', 'r', 's', 'a', 'c', 'z'}
setA = setA | setB | setC
print(setA) # {abcde,pqrs,xyz}

{'r', 'p', 'a', 'q', 's', 'd', 'e', 'x', 'b', 'z', 'c', 'y'}

```

```

# Shallow Copy
# List -> .copy() , [:] Slicing , list() constructor
# set -> .copy() , set() constructor
# Copying the Set() : Performing Shallow Copy
day_set = {'Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun'}
copy_day_set = day_set.copy() # shallow Copy
print(id(day_set))
print(id(copy_day_set)) # Different Memory Address

2252425832128
2252425837728

# Set() is a mutable Container that stores only immutable elements
copy_day_set.add('Jan')
print(day_set)
print(copy_day_set) # 'Jan'

```

```

print(day_set)
print(copy_day_set) # 'Jan'
{'Sat', 'Sun', 'Fri', 'Thur', 'Wed', 'Tue', 'Mon'}
{'Sat', 'Sun', 'Fri', 'Thur', 'Wed', 'Tue', 'Jan', 'Mon'}

```

```

# Copying the set using set() constructor
another_day_set = set(copy_day_set) # Shallow Copy
print(id(copy_day_set)) # 2252425837728
print(id(another_day_set)) # Different Memory Address
another_day_set.discard('Jan')
print(another_day_set) # {'Sat', 'Sun', 'Fri', 'Thur', 'Wed', 'Tue', 'Mon'}
print(copy_day_set) # {'Sat', 'Sun', 'Fri', 'Thur', 'Wed', 'Tue', 'Jan', 'Mon'}
2252425837728
2252425835264
{'Sat', 'Wed', 'Tue', 'Mon', 'Sun', 'Fri', 'Thur'}
{'Sat', 'Sun', 'Fri', 'Thur', 'Wed', 'Tue', 'Jan', 'Mon'}

```

What is Dictionary in Python?

A Dictionary is :

1. **Ordered** : Items have guaranteed sequence.
2. **Mutable** : You can add,remove or change items.
3. **Collection of Key-Value Pair** : Each key is unique and maps to a value.

```

_dict = {} # empty curly braces represent dictionary class not a set()
print(_dict)
print(type(_dict))

{}
<class 'dict'>

```

```

# Dictionary : # When defining duplicate keys, only the last occurrence is kept.
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python'],
    'course' : 'Data Science'
}
student_details

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'course': 'Data Science',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}

```

```

# Key -> Unique [Tuple] # As its a immutable container
_dict = {
    ('stud1','stud2') : 98,
    ('stud3','stud4') : 92,
    ('stud5','stud6') : 77,
    ('stud7','stud8') : 99
}
_dict

{('stud1', 'stud2'): 98,
 ('stud3', 'stud4'): 92,
 ('stud5', 'stud6'): 77,
 ('stud7', 'stud8'): 99}

# dict() Constructor
_dict = dict(name = 'Shyam Sundar', age = 27, gender = 'Male' , course = 'Data Analytics')
_dict

{'name': 'Shyam Sundar',
 'age': 27,
 'gender': 'Male',
 'course': 'Data Analytics'}

```

```

print(type(_dict)) # 'dict'
<class 'dict'>

# List of List as a Pair -> Dict # Pairing(2 elements)
list_of_lists = [
    ['name','Nishant'],
    ['age',25],
    ['gender','Male'],
    ['city','Ranikhet'],
    ['state','Uttarakhand'],
    ['country','India']
]
_dict = dict(list_of_lists)
_dict

{'name': 'Nishant',
 'age': 25,
 'gender': 'Male',
 'city': 'Ranikhet',
 'state': 'Uttarakhand',
 'country': 'India'}

```

```

# List of Tuples as a Pair -> Dict # Pairing(2 elements)
list_of_tuples = [
    ('name','Nishant'),
    ('age',25),
    ('gender','Male'),
    ('city','Ranikhet'),
    ('state','Uttarakhand'),
    ('country','India')
]
_dict = dict(list_of_tuples)
_dict

{'name': 'Nishant',
 'age': 25,
 'gender': 'Male',
 'city': 'Ranikhet',
 'state': 'Uttarakhand',
 'country': 'India'}

```

```
'city': 'Ranikhet',
'state': 'Uttarakhand',
'country': 'India'}
```

```
# Tuple of Tuples as a Pair -> Dict # Pairing(2 elements)
tuple_of_tuples = (
    ('name', 'Nishant'),
    ('age', 25),
    ('gender', 'Male'),
    ('city', 'Ranikhet'),
    ('state', 'Uttarakhand'),
    ('country', 'India')
)
_dict = dict(tuple_of_tuples)
_dict

{'name': 'Nishant',
 'age': 25,
 'gender': 'Male',
 'city': 'Ranikhet',
 'state': 'Uttarakhand',
 'country': 'India'}
```

```
# Tuple of Lists as a Pair -> Dict # Pairing(2 elements)
tuple_of_lists = (
    ['name', 'Nishant'],
    ['age', 25],
    ['gender', 'Male'],
    ['city', 'Ranikhet'],
    ['state', 'Uttarakhand'],
    ['country', 'India']
)
_dict = dict(tuple_of_lists)
_dict

{'name': 'Nishant',
 'age': 25,
 'gender': 'Male',
 'city': 'Ranikhet',
 'state': 'Uttarakhand',
 'country': 'India'}
```

```
# List of Sets as a Pair -> Dict # Pairing(2 elements) ✗ NOT Recommended ✗
list_of_sets = [
    {'name', 'Nishant'},
    {'age', 25},
    {'gender', 'Male'},
    {'city', 'Ranikhet'},
    {'state', 'Uttarakhand'},
    {'country', 'India'}
]
_dict = dict(list_of_sets)
_dict # {key-value pair will be shuffled}

{'name': 'Nishant',
 'age': 25,
 'gender': 'Male',
 'city': 'Ranikhet',
 'state': 'Uttarakhand',
 'India': 'country'}
```

```

# Tuple of Lists as a Pair -> Dict # Pairing( Exactly 2 elements) [2 arguments]
# Make the 2nd Argument [Iterable Container]
tuple_of_lists = (
    ['name', 'Nishant'],
    ['age', 25],
    ['gender', 'Male'],
    ['city', 'Ranikhet'],
    ['state', 'Uttarakhand'],
    ['country', 'India'],
    ['skills', ['Python', 'SQL']])
)
_dict = dict(tuple_of_lists)
_dict

{'name': 'Nishant',
 'age': 25,
 'gender': 'Male',
 'city': 'Ranikhet',
 'state': 'Uttarakhand',
 'country': 'India',
 'skills': ['Python', 'SQL']}

```

```

_list = [[11, 'Rupees']]
_dict = dict(_list)
_dict

{11: 'Rupees'}

_dict = {
    98 : ('stud1', 'stud2'),
    92 : ('stud3', 'stud4'),
    77 : ('stud5', 'stud6'),
    99 : ('stud7', 'stud8'),
    99 : ('stud9', 'stud10'),
}
_dict

{98: ('stud1', 'stud2'),
 92: ('stud3', 'stud4'),
 77: ('stud5', 'stud6'),
 99: ('stud9', 'stud10')}

```

```

student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel', 'PowerBI', 'MySQL', 'Python'],
    'course' : 'Data Science'
}
student_details

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'course': 'Data Science',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}

```

```

# Nested Dictionaries:
student_record = {
    "Student1": {
        'name' : 'Rajat Singh Thakur',
        'age' : 29,
        'gender' : 'Male',
        'city' : 'Shimla',
        'state' : 'Himachal Pradesh',
        'country' : 'India',
        'course' : 'Data Analytics',
        'skills' : ['Excel','PowerBI','MySQL','Python']
    },
    "Student2": {
        'name' : 'Shyam Sundar',
        'age' : 25,
        'gender' : 'Male',
        'city' : 'Vizag',
        'state' : 'Andhra Pradesh',
        'country' : 'India',
        'course' : 'Data Analytics',
        'skills' : ['Excel','PowerBI','MySQL','Python']
    }
}
student_record

```

```

{'Student1': {'name': 'Rajat Singh Thakur',
  'age': 29,
  'gender': 'Male',
  'city': 'Shimla',
  'state': 'Himachal Pradesh',
  'country': 'India',
  'course': 'Data Analytics',
  'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']},
 'Student2': {'name': 'Shyam Sundar',
  'age': 25,
  'gender': 'Male',
  'city': 'Vizag',
  'state': 'Andhra Pradesh',
  'country': 'India',
  'course': 'Data Analytics',
  'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}}

```

```

# Common Methods and Operations Associated with Dictionaries
# Length() -> Use Len() to find how many key-value pairs available in a dict.
print(len(student_details)) # 8
8

print(len(student_record)) # 2
2

print(len(student_record['Student2'])) # 8
8

```

```
# Bracket Notation is used here to apply a filter by passing a key
student_record['Student2']

{'name': 'Shyam Sundar',
 'age': 25,
 'gender': 'Male',
 'city': 'Vizag',
 'state': 'Andhra Pradesh',
 'country': 'India',
 'course': 'Data Analytics',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}
```

```
# Accessing an Element in Dictionaries
# Use Square Bracket [Key]
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python'],
    'course' : 'Data Science'
}
student_details
student_details['skills'] # ['Excel', 'PowerBI', 'MySQL', 'Python']
['Excel', 'PowerBI', 'MySQL', 'Python']

student_details['course'] # 'Data Science'

'Data Science'
```

```
# Nested Dictionaries:
student_record = {
    "Student1": {
        'name' : 'Rajat Singh Thakur',
        'age' : 29,
        'gender' : 'Male',
        'city' : 'Shimla',
        'state' : 'Himachal Pradesh',
        'country' : 'India',
        'course' : 'Data Analytics',
        'skills' : ['Excel','PowerBI','MySQL','Python']
    },
    "Student2": {
        'name' : 'Shyam Sundar',
        'age' : 25,
        'gender' : 'Male',
        'city' : 'Vizag',
        'state' : 'Andhra Pradesh',
        'country' : 'India',
        'course' : 'Data Analytics',
        'skills' : ['Excel','PowerBI','MySQL','Python']
    }
}
student_record['Student2']['name'] # 'Shyam Sundar'
```

```

'Shyam Sundar'

student_record['Student2']['skills'] # ['Excel', 'PowerBI', 'MySQL', 'Python']
['Excel', 'PowerBI', 'MySQL', 'Python']

student_record['Student2']['skills'][-1] # 'Python'
'Python'

student_record['Student2']['skills'][2:] # ['MySQL', 'Python']
['MySQL', 'Python']

student_record['Student7'] # KeyError: 'Student7'

```

```

# Key -> Unique [Tuple] # As its a immutable container
_dict = {
    ('stud1','stud2') : 98,
    ('stud3','stud4') : 92,
    ('stud5','stud6') : 77,
    ('stud7','stud8') : 99
}
_dict[('stud5','stud6')] # 77
77

```

```

# .get(key, default) method -> It will safely retrieve the value.
# Returns Default if no key exist.
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel', 'PowerBI', 'MySQL', 'Python']
}
student_details.get('course', 'Data Science')

'Data Analytics'

student_details.get('email', 'xyz@gmail.com')

'xyz@gmail.com'

student_details.get('skills') # ['Excel', 'PowerBI', 'MySQL', 'Python']
['Excel', 'PowerBI', 'MySQL', 'Python']

```

```

student_details.get('phone_number') # Won't throw an error

# Nested Dictionaries:
student_record = {
    "Student1": {
        'name' : 'Rajat Singh Thakur',
        'age' : 29,
        'gender' : 'Male',
        'city' : 'Shimla',
        'state' : 'Himachal Pradesh',
        'country' : 'India',
        'course' : 'Data Analytics',
        'skills' : ['Excel','PowerBI','MySQL','Python']
    },
    "Student2": {
        'name' : 'Shyam Sundar',
        'age' : 25,
        'gender' : 'Male',
        'city' : 'Vizag',
        'state' : 'Andhra Pradesh',
        'country' : 'India',
        'course' : 'Data Analytics',
        'skills' : ['Excel','PowerBI','MySQL','Python']
    }
}
student_record.get('Student2' , {}).get('name','Unknown') # 'Shyam Sundar'

```

```

student_record.get('Student1' , {})

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'course': 'Data Analytics',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}

```

```

student_record.get('Student7' , {}).get('name','Unknown') # 'Unknown'

'Unknown'

```

```

student_record.get('Student7' , {})

{}

```

```

# .keys() -> Returns all keys available in a dictionary
# .values() -> Returns all values available in a dictionary
# .items() -> Returns all [key,value] pair available in a dictionary
student_details.keys()

dict_keys(['name', 'age', 'gender', 'city', 'state', 'country', 'course', 'skills'])

student_record.keys()

dict_keys(['Student1', 'Student2'])

student_record['Student2'].keys()

dict_keys(['name', 'age', 'gender', 'city', 'state', 'country', 'course', 'skills'])

student_record.get('Student1', {}).keys()

dict_keys(['name', 'age', 'gender', 'city', 'state', 'country', 'course', 'skills'])

student_record.get('Student7', {}).keys()

dict_keys([])

```

```

student_details.values()

dict_values(['Rajat Singh Thakur', 29, 'Male', 'Shimla', 'Himachal Pradesh', 'India', 'Data Analytics', ['Excel', 'PowerBI', 'MySQL', 'Python']])

student_record.values()

dict_values([{'name': 'Rajat Singh Thakur', 'age': 29, 'gender': 'Male', 'city': 'Shimla', 'state': 'Himachal Pradesh', 'country': 'India', 'course': 'Data Analytics', 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}, {'name': 'Shyam Sundar', 'age': 25, 'gender': 'Male', 'city': 'Vizag', 'state': 'Andhra Pradesh', 'country': 'India', 'course': 'Data Analytics', 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}])

student_record['Student2'].values()

dict_values(['Shyam Sundar', 25, 'Male', 'Vizag', 'Andhra Pradesh', 'India', 'Data Analytics', ['Excel', 'PowerBI', 'MySQL', 'Python']])

student_record.get('Student1', {}).values()

dict_values(['Rajat Singh Thakur', 29, 'Male', 'Shimla', 'Himachal Pradesh', 'India', 'Data Analytics', ['Excel', 'PowerBI', 'MySQL', 'Python']])

student_record.get('Student7', {}).values()

dict_values([])

```

```

student_details.items()

dict_items([('name', 'Rajat Singh Thakur'), ('age', 29), ('gender', 'Male'), ('city', 'Shimla'), ('state', 'Himachal Pradesh'), ('country', 'India'), ('course', 'Data Analytics'), ('skills', ['Excel', 'PowerBI', 'MySQL', 'Python'])])

# setdefault(key,[default]) -> Retrieves the value if the key exists,
# else it would be inserting key with default.
# if no default provided , update the value with None.

student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python']
}
student_details.setdefault('course') # 'Data Analytics'

'Data Analytics'

```

```

student_details.setdefault('course' , 'Data Science') # 'Data Analytics'

'Data Analytics'

student_details.setdefault('email' , 'xyz@gmail.com') # 'xyz@gmail.com'

'xyz@gmail.com'

student_details

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'course': 'Data Analytics',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python'],
 'email': 'xyz@gmail.com'}

student_details.setdefault('phone_number') # 'None'

print(student_details.setdefault('phone_number')) # 'None'

None

```

```

student_details

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'course': 'Data Analytics',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python'],
 'email': 'xyz@gmail.com',
 'phone_number': None}

```

```

# .min() , .max() , .sum()
# Key -> Unique [Tuple] # As its a immutable container
_dict = {
    ('stud1','stud2') : 98,
    ('stud3','stud4') : 92,
    ('stud5','stud6') : 77,
    ('stud7','stud8') : 99
}
min(_dict.values()) # [98,92,77,99] # 77
77

max(_dict.values()) # [98,92,77,99] # 99
99

sum(_dict.values()) # [98,92,77,99] # 366
366

```

```

min(_dict.keys()) # [('stud1','stud2')]
('stud1', 'stud2')

max(_dict.keys()) # [('stud7','stud8')]
('stud7', 'stud8')

_dict = {
    98 : ('stud1','stud2'),
    92 : ('stud3','stud4'),
    77 : ('stud5','stud6'),
    99 : ('stud7','stud8'),
    99 : ('stud9','stud10'),
}
_dict
min(_dict.keys()) # 77
77

max(_dict.keys()) # 99
99

sum(_dict.keys()) # 366
366

```

```
# changing or adding a Dictionary Items
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python']
}
student_details['course'] = 'Data Science'

student_details
```

```
student_details['email'] = 'rajat.thakur_007@gmail.com'
student_details
```

```
# .update()
stud_info = {'name' : 'Bhupinder Jogi' , 'age' : 40 , 'city' : 'Lukhnow' , 'state' : 'Uttar Pradesh' ,
             'country' : 'America' , 'email' : 'bhupinder.jogi@lallantop.in'}

student_details.update(stud_info)
student_details
```

```
student_record.get('Student1' , {}).update(stud_info)
student_record['Student1']

{'name': 'Bhupinder Jogi',
 'age': 40,
 'gender': 'Male',
 'city': 'Lukhnow',
 'state': 'Uttar Pradesh',
 'country': 'America',
 'course': 'Data Analytics',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python'],
 'email': 'bhupinder.jogi@lallantop.in'}
```

```
student_record.get('Student7' , {}).update(stud_info) # Doesn't do anything
student_record

{'Student1': {'name': 'Bhupinder Jogi',
  'age': 40,
  'gender': 'Male',
  'city': 'Lukhnow',
  'state': 'Uttar Pradesh',
  'country': 'America',
  'course': 'Data Analytics',
  'skills': ['Excel', 'PowerBI', 'MySQL', 'Python'],
  'email': 'bhupinder.jogi@lallantop.in'},
 'Student2': {'name': 'Shyam Sundar',
  'age': 25,
  'gender': 'Male',
  'city': 'Vizag',
  'state': 'Andhra Pradesh',
  'country': 'India',
  'course': 'Data Analytics',
  'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}}}
```

```
# .zip() -> Combine 2 iterables into Key-Value Pair
key = ['name', 'age', 'city', 'gender', 'country']
value = ('Deepak', 24, 'Delhi', 'Male', 'India')
_zip = zip(key,value)
_zip

<zip at 0x20c705d6780>

# Dict() Constructor
zip_dict = dict(_zip)
zip_dict

{'name': 'Deepak',
 'age': 24,
 'city': 'Delhi',
 'gender': 'Male',
 'country': 'India'}
```

```
# Removing a Dictionary Items
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh'
```

```

        'gender' : 'Male',
        'city' : 'Shimla',
        'state' : 'Himachal Pradesh',
        'country' : 'India',
        'course' : 'Data Analytics',
        'skills' : ['Excel','PowerBI','MySQL','Python']
    }
del student_details['course']
student_details

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'skills': ['Excel', 'PowerBI', 'MySQL', 'Python']}

```

```

# Removing a Dictionary Items
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python']
}
del student_details # Delete the entire Dictionary

student_details # NameError: name 'student_details' is not defined

```

```

# Removing a Dictionary Items
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python']
}
del student_details['skills'][0] # ['Excel']
student_details

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'course': 'Data Analytics',
 'skills': ['PowerBI', 'MySQL', 'Python']}

```

```
# pop() -> Removes a specific key and returns its values
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python']
}
skill_info = student_details.pop('skills')
skill_info # ['Excel', 'PowerBI', 'MySQL', 'Python']

['Excel', 'PowerBI', 'MySQL', 'Python']

student_details

{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India',
 'course': 'Data Analytics'}
```

```
# pop() -> Removes a specific key and returns its values
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python']
}
email = student_details.pop('email') # KeyError: 'email'

# popitem() # -> Removes the last inserted Values
course_detail = student_details.popitem()
course_detail

('course', 'Data Analytics')
```

```

student_details
{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh',
 'country': 'India'}

# popitem() # -> Removes the last inserted Values
country_detail = student_details.popitem()
country_detail

('country', 'India')

student_details
{'name': 'Rajat Singh Thakur',
 'age': 29,
 'gender': 'Male',
 'city': 'Shimla',
 'state': 'Himachal Pradesh'}

```

```

# .clear() -> Empties the Dictionary
student_details.clear()
student_details # {}

{}

# Shallow Copy -> Dict ..... [.copy() , dict() Constructor]
student_details = {
    'name' : 'Rajat Singh Thakur',
    'age' : 29,
    'gender' : 'Male',
    'city' : 'Shimla',
    'state' : 'Himachal Pradesh',
    'country' : 'India',
    'course' : 'Data Analytics',
    'skills' : ['Excel','PowerBI','MySQL','Python']
}
stud_details = student_details.copy() # Shallow Copy
print(id(stud_details))
print(id(student_details)) # Different Memory Locations

2252447691584
2252447680000

```

```

# Shallow Copy -> using dict() Constructor
shallow_dict = dict(stud_details)
print(id(shallow_dict)) # Different Memory Address
print(id(stud_details)) # 2252447691584

2252447600064
2252447691584

```