Day 3 of learning python set,dictionary

1. Create a set with the first five prime numbers

```
prime_number={2,3,5,7,11}
print(prime_number)
```

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

Add the number 7 to the set.

```
{2,3,5,7,11}.union({11})
```

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

. Remove the number 3 from the set

```
prime_number={2,3,5,7,11}
remove={3}
prime_number-remove
```

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

Check if the set is a subset of {2, 3, 5, 7, 11}.

```
prime_number={2, 5, 7, 11}
prime_number.issubset({2, 3, 5, 7, 11})
```

```
True
```

Find the union of the set with {7, 11, 13}.

```
prime_number={2, 3, 5, 7, 11}
seet={7,11,13}
prime_number.union(seet)
```

```
{2, 3, 5, 7, 11, 13}
```

Create a frozen set from the original set.

```
s=frozenset({"abkab","ee"})
print(s.remove("ee"))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-23-8154fd796526> in <cell line: 2>()
      1 s=frozenset({"abkab","ee"})
----> 2 print(s.remove("ee"))
      3

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

Check if the set has any common elements with {1, 4, 9}

```
prime_number={2, 3, 5, 7, 11}
prime_number.intersection({1,4,9})
```

```
set()
```

Remove all elements from the set.

Double-click (or enter) to edit

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

prime_number.clear()
```

. Create a set of your favorite fruits and nd the intersection with { 'apple', 'banana', 'orange' }.

```
friuts={"mango","banana"}
fruits2={ 'apple', 'banana', 'orange' }
friuts.intersection(fruits2)
```

```
{'banana'}
```

Use set comprehension to create a set of squares for numbers 1 to 10.

Double-click (or enter) to edit

```
squares_set = {x**2 for x in range(1, 11)}

# Printing the set of squares
print(squares_set)
```

```
{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
```

list . Create a list of integers from 1 to 5

```
se=[x for x in range(1,6)]
print(se)
```

```
[1, 2, 3, 4, 5]
```

. Append the number 6 to the list.

```
se=[1, 2, 3, 4, 5]
se.append(6)
print(se)
```

```
[1, 2, 3, 4, 5, 6]
```

3. Extend the list with another list: [7, 8, 9].

```
se=[1, 2, 3, 4, 5,6]
dr=  [7, 8, 9]
se.extend(dr)
print(se)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

. Access and print the third element of the list

Double-click (or enter) to edit

```
se=[1, 2, 3, 4, 5, 6, 7, 8, 9]
se[2]
```

```
3
```

1. Basic Dictionary Operations: Create a dictionary named appropriate values. student with keys "name," "age," and "grade," and assign Print the value associated with the "age" key. Update the "grade" to a new value. Add a new key-value pair for "subject" and its corresponding value.

```
student={'name':"nitin",'age':20,'grade':100}
print(student.get('age'))
students2={'grade':200}
student.update(students2)
print(student)
student['subject']='hindi'
print(student)
```

```
20
{'name': 'nitin', 'age': 20, 'grade': 200}
{'name': 'nitin', 'age': 20, 'grade': 200, 'subject': 'hindi'}
```

Dictionary Manipulation: Create two dictionaries, dict1 and dict2, with at least three key-value pairs each. Merge these dictionaries into a new dictionary called merged_dict. Remove a key from merged_dict. Check if a speci c key exists in dict1

```
dic1={'name':"nitin",'age':20,'grade':100}
dic2={'fruit':"apple",'mobile':"iphone",'laptop':"asus"}
merge_dic={**dic1,**dic2}
print(merge_dic)
merge_dic.pop('fruit')
print(merge_dic)
check='name'
check_in_dic1=check in dic1
print("check a name key in dic ?\n",check_in_dic1)
```

```
{'name': 'nitin', 'age': 20, 'grade': 100, 'fruit': 'apple', 'mobile': 'iphone', 'laptop': 'asus'}
{'name': 'nitin', 'age': 20, 'grade': 100, 'mobile': 'iphone', 'laptop': 'asus'}
check a name key in dic ?
 True
```

3. Iterating Through a Dictionary: Use a loop to print all keys in the student dictionary. Use another loop to print all values in the student dictionary. Write a loop to print each key-value pair in the

```
my_dict = {'1': 'MUKTA', '2': 'DIKSHA', '3': 'NISHTHA'}
print("Keys in my_dict dictionary:")
for key in my_dict.keys():
    print(key)
print("values in my_dict dictionary:")
for values in my_dict.values():
    print(values)
print("key pair in my_dic Dictionary")

for key,values in student.items():
    print(f"{key}: {values}")
```

```
Keys in my_dict dictionary:
1
2
3
values in my_dict dictionary:
MUKTA
DIKSHA
NISHTHA
key pair in my_dic Dictionary
name: nitin
age: 20
grade: 200
subject: hindi
```

Create a dictionary comprehension to generate a dictionary of squares from 1 to 10. Filter the above dictionary to include only even squares.

```
squares_dict = {num: num**2 for num in range(1, 11)}
print("Dictionary of squares from 1 to 10:")
print(squares_dict)
even_squares_dict = {num: num**2 for num in range(1, 11) if num % 2 == 0}
print("Filtered dictionary with only even numbers and their squares:")
print(even_squares_dict )
```

```
Dictionary of squares from 1 to 10:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
Filtered dictionary with only even numbers and their squares:
{2: 4, 4: 16, 6: 36, 8: 64, 10: 100}
```

Nested Dictionaries: Create a dictionary named school with multiple students (nested dictionaries). Access and print information about a speci c student within the school dictionary

```
school = {
    "student1": {
        "name": "Alice Johnson",
        "age": 14,
        "grade": 9
    },
    "student2": {
        "name": "Bob Smith",
        "age": 15,
        "grade": 10
    },
    "student3": {
        "name": "Charlie Brown",
        "age": 16,
        "grade": 11
    }
}


student_id = "student2"
if student_id in school:
    student_info = school[student_id]
    print(f"Information for {student_id}:")
    print(f"Name: {student_info['name']}")
    print(f"Age: {student_info['age']}")
    print(f"Grade: {student_info['grade']}")
else:
    print(f"No information found for {student_id}.")
```

```
Information for student2:
Name: Bob Smith
Age: 15
Grade: 10
```

. Dictionary Functions: Use the len() function to nd the number of items in a dictionary. Use the keys(), values(), and items() methods on a dictionary and print the results.

```
dic1={'1': 'MUKTA', '2': 'DIKSHA', '3': 'NISHTHA'}
print(len(dic1))
print(dic1.keys())
print(dic1.values())
```

```
3
dict_keys(['1', '2', '3'])
dict_values(['MUKTA', 'DIKSHA', 'NISHTHA'])
```

Create a dictionary with unsorted keys. Use the sorted() function to print the keys in alphabetical order.

```
dic1={'name':"nitin",'age':20,'grade':100}
print(sorted(dic1))
```

```
['age', 'grade', 'name']
```

Use the get() method to retrieve a value with a default if the key doesn't exist. Use the pop() method to remove and return a speci c key-value pair

```
dic1={'name':"nitin",'age':20,'grade':100}
print(dic1.get('ki'))
dic1.pop('age')
print(dic1)
```

```
None
{'name': 'nitin', 'grade': 100}
```

4. Dictionary Operations: Create a dictionary named book with keys "title," "author," and "year," and assign appropriate values. Print the author of the book. Update the year of the book to a new value. Remove the "title" key from the dictionary

```
book={'title':"freedom",'author':"ram_singh",'year':2000}
print(book.get('author'))
book['year']=1980
(book.pop('title'))
print(book)
```

```
ram_singh
{'author': 'ram_singh', 'year': 1980}
```

Create a dictionary named library with multiple books (nested dictionaries). Access and print information about a speci c book within the library information .

```
library={'book1':{'title':"freedom",'author':"ram_singh",'year':2000},'book2':{'title':"dust",'author':"ram",'year':200},'book3':{'title':"right
book_id="book1"
if book_id in library:
  book_info=library[book_id]
  print(book_info)
```

```
{'title': 'freedom', 'author': 'ram_singh', 'year': 2000}
```

1. Set Operations: Create two sets, set1 and set2, with some common elements. Find and print the union of the two sets. Determine the intersection of the sets. Check if one set is a subset of the other

```
set1={1,2,3,4,5}
set2={3,4,5,6,}
print(set1.union(set2))
print(set1.intersection(set2))
set1.issubset(set2)
```

```
{1, 2, 3, 4, 5, 6}
{3, 4, 5}
False
```

2. Set Manipulation: Add an element to set1. Remove an element from set2. Clear all elements from one of the sets.

```
set1={1,2,3,4,5}
set2={3,4,5,6,}
set1.add(6)
print(set1)
(set2.remove(4))
print(set2)
```

```
{1, 2, 3, 4, 5, 6}
{3, 5, 6}
```

Double-click (or enter) to edit

Double-click (or enter) to edit

3. Set Comprehension: Create a set comprehension that generates a set of squares from 1 to 10. Filter the above set to include only even squares

Double-click (or enter) to edit

```
squares_set = {num**2 for num in range(1, 11)}
print("Dictionary of squares from 1 to 10:")
print(squares_set)
even_squares_set = {num**2 for num in range(1, 11) if num % 2 == 0}
print("Filtered set with only even numbers their squares:")
print(even_squares_set )
```

```
Dictionary of squares from 1 to 10:
{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
Filtered set with only even numbers their squares:
{64, 100, 4, 36, 16}
```

Double-click (or enter) to edit

List Operations: Create a list named

fruits with at least ve different fruit names. Add a new fruit to the list. Remove a speci c fruit from the list. Reverse the order of the elements in the list

```
list1=['apple','mango','orange','banana','berry']
print(list1)
list1.remove('apple')
print(list1)
list1.reverse()
print(list1)
```

```
['apple', 'mango', 'orange', 'banana', 'berry']
['mango', 'orange', 'banana', 'berry']
['berry', 'banana', 'orange', 'mango']
```

7. List Comprehension:

   Create a list comprehension that generates a list of cubes from 1 to 10. Filter the above list to include only even cubes.

```
lsit1=[x**3 for x in range(1,11)]
print(lsit1)
lsit1=[x**3 for x in range(1,11)if x%2==0]
print("only cubes in this range",lsit1)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
only cubes in this range [8, 64, 216, 512, 1000]
```

Tuples: . Tuple Operations: Create a tuple named colors with at least three different color names. Access and print the second element of the tuple. Try to change the value of one element in the tuple and explain the result.

```
tuple1=('red','green','orange')
print(tuple1[1])
tuple1[1]='yellow'
print(tuple1)
```

```
green
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-61-5dae3846e951> in <cell line: 3>()
      1 tuple1=('red','green','orange')
      2 print(tuple1[1])
----> 3 tuple1[1]='yellow'
      4 print(tuple1)

TypeError: 'tuple' object does not support item assignment
```

9. Mixed Sequences: Create a list that contains a mix of sets, dictionaries, lists, and tuples. Access and print an element from each of these sequences within the list

```
mixed_list = [
    {1, 2, 3},                   # Set
    {'key1': 'value1', 'key2': 'value2'},  # Dictionary
    [4, 5, 6],                   # List
    ('a', 'b', 'c')              # Tuple
]
set_element = list(mixed_list[0])[0]
print(f"Element from set: {set_element}")
dict_element = mixed_list[1]['key1']
print(f"Element from dictionary: {dict_element}")
list_element = mixed_list[2][1]
print(f"Element from list: {list_element}")
tuple_element = mixed_list[3][2]
print(f"Element from tuple: {tuple_element}")
```

```
Element from set: 1
Element from dictionary: value1
Element from list: 5
Element from tuple: c
```