```python
#Tuples : immutable , ordered
# similar to list

# gps coordinates (latitude, laongitude )

# created using parenthesiis

my_tuple = (1,2,4,5)
print(my_tuple)

mytuple = (1,"Anna" ,True, 3002)
print(mytuple)

print(type(mytuple))
print(mytuple[0])
print(mytuple[-1])

#Slicing a tupke

print(mytuple[1:4])

#Unpack allows assigning tuple elements to multiple variables
id , name , isEmployed , salary = mytuple
print(name)
print(isEmployed)

for ele in mytuple:
    print(ele)

# Built in methods
#count() -- > count the occurences
tuple_numbers = (1,2,3,2,1,4,2,1)
print(tuple_numbers.count(1))


#index --> first occurencce
print(tuple_numbers.index(1))


# tuple_numbers[0] = 100 --> will give type error

# tuples can be nested
my_tuple = (1,(2,3),(4,5))
```

```
print(my_tuple)

lst = list(my_tuple)
print(lst)

tuple1 = tuple(lst)
print(tuple1)

# faster than list  , consume less memory
{
    (): "value"
}



# Remove an empty tuple(s) from a list of tuples
Lsttuples = [(),(),(),(''),('a','b'),('a','b','c')]
removeemppty = [t for t in Lsttuples if t]
print(removeemppty)

#compare two tuples lexiographically
a = (1,2,3)
b = (1,2,4)
print(a < b)

# sum built in method
print(sum(a))

# reverse a tuple
reversed_tuple = tuple(reversed(a))
print(reversed_tuple)

# max and min
print(max(a))
```

Create a tuple with different data types and print it.
Access the second element in a tuple.
Try modifying a tuple element and observe the error.
Convert a list to a tuple.
Find the length of a tuple.
Concatenate two tuples.
Use + operator
Repeat a tuple 3 times.
Check if an element exists in a tuple.
Unpack a tuple into variables.
Slice a tuple from index 1 to 3.
Find the index of an element in a tuple.
Count the occurrences of an element in a tuple.
Convert a tuple into a string.
Create a nested tuple.
Iterate through a tuple using a loop.
Find the maximum and minimum values in a numeric tuple.
Reverse a tuple.
Find the sum of elements in a numeric tuple.
Convert a tuple of tuples into a dictionary.
Compare two tuples lexicographically.

SETS

```
# Python sets
# no duplicate values
# mutable(can add/remove elements)
#undordered --> do not have definite order

my_set = {1 , 3,1,5}
print(my_set)
print(type(my_set))

# set methods

#add
my_set.add(4)
print(my_set)

my_set.update([5,6,7])
print(my_set)
```

```python
# my_set.remove(10)  # throw key error
my_set.discard(10)  # does not throw error
print(my_set)
#pop

ele = my_set.pop()
print(f"removed : {ele} , Remaining set {my_set}")



my_set.clear()
print(my_set)

my_dict = {}
print(my_dict)


# Set Operations
# union --> combines (OR | operator)

a= {1,2,3}
b = {2,3,6}
print(a.union(b))
print(a|b)
print(a)
print(b)

# Intersection
print(a.intersection(b))
print(a & b)


# differnece
print(a-b)
print(a.difference(b))

# subset
# issubset()
print(a.issubset(b))
# print(a.isproperset(b))

c = {2,3,4}
```

```
d = {6}
print(c.issuperset(d))


# isdisjoint() -- > checks if two sets have no common elements

print(c.isdisjoint(d))

print(a.symmetric_difference(b))


a= {2,3}
b = {2,3}

def is_proper_set(a,b):
    return b.issubset(a) and a!=b

print(is_proper_set(a,b))


# copy() creates copy

copy_set = c.copy()
print(copy_set)

# Frozen sets (immutable)
frozen = frozenset([1,2,3])
frozen.add(5)

a.update(b)
print(a)
```

1. **Create a set**
   Create a set with elements `{1, 2, 3, 4, 5}` and print it.
2. **Add an element**
   Add the element `6` to the set and print the updated set.
3. **Remove an element**
   Remove the element `3` from the set and print the updated set.
4. **Discard an element (no error if not present)**
   Discard the element `10` from the set and print the updated set.

5. **Check if elements exist**
   Check if elements `3` and `6` exist in the set and print the results.
6. **Find the length of the set**
   Create a set `{1, 2, 3, 4, 5, 5, 6}`, print its length, and observe how duplicates are handled.
7. **Set difference**
   Find the difference between sets `{1, 2, 3, 4}` and `{3, 4, 5, 6}` and print the result.
8. **Set union**
   Find the union of sets `{1, 2, 3, 4}` and `{3, 4, 5, 6}` and print the result.
9. **Set intersection**
   Find the intersection of sets `{1, 2, 3, 4}` and `{3, 4, 5, 6}` and print the result.
10. **Symmetric difference**
    Find the symmetric difference between sets `{1, 2, 3, 4}` and `{3, 4, 5, 6}` and print the result.
11. **Subset check**
    Check if `{1, 2}` is a subset of `{1, 2, 3, 4}` and print the result.
12. **Superset check**
    Check if `{1, 2, 3, 4}` is a superset of `{1, 2}` and print the result.
13. **Disjoint sets check**
    Check if sets `{1, 2, 3}` and `{4, 5, 6}` are disjoint and print the result.
14. **Copy a set**
    Copy set `{1, 2, 3, 4}` into another variable and check if they are different objects.
15. **Pop an element**
    Remove a random element from the set `{10, 20, 30, 40, 50}` using `pop()` and print the removed element.
16. **Clear a set**
    Clear all elements from a set and print the empty set.
17. **Update a set**
    Update the set `{1, 2, 3}` with elements `{4, 5, 6}` and print the updated set.
18. **Difference update**
    Remove elements from `{1, 2, 3, 4, 5}` that are also present in `{3, 4, 5, 6, 7}` using `difference_update()`.
19. **Intersection update**
    Retain only elements in `{1, 2, 3, 4, 5}` that are also in `{3, 4, 5, 6, 7}` using `intersection_update()`.
20. **Symmetric difference update**
    Update set `{1, 2, 3}` to contain only elements that are in either `{1, 2, 3}` or `{2, 3, 4}` but not both.

4o