comparison between 4 collections in Python i.e., Tuples, List, set, Dictionary in tabular column and small examples for such

→ Python has 4 built-in data structures that can be used to hold a collection of objects, they are list, tuple, set and dictionary. They can be distinguished into mutable, immutable, set types, and mappings respectively.

List:-

→ List are ordered mutable sequence that can be changed after they have been created by adding, removing, (or) changing objects.

→ List can be declared by using square brackets "[]" following variable name.

Tuples:-

→ Tuples are ordered immutable sequence that stores multiple items in a single variable, meaning it cannot be changed after it has been created.

→ A tuple can be created by a pair of parenthesis and common-separated objects, following the variable name.

set:-

→ set are an unordered immutable set of unique objects that do not support duplicated objects and as such, they cannot be indexed.

→ mathematical operations such as intersection, union, difference, and symmetric difference, can be carried out on set data types.

Dictionary:

→ A dictionary is an unordered set of key/value pairs.
→ each unique key has a value associated with it in the dictionary, and dictionaries can have any number of pairs.

Representing in tabular column

Data structure	Definition	Mutable	ordered	Indexable	Allows duplicates	example
Tuples	ordered, immutable collection	NO	Yes	Yes	Yes	mytuple = (1, 2, 3)
Lists	ordered, mutable collection	Yes	Yes	Yes	Yes	mylist = [1, 2, 3]
sets	unordered, mutable, collection	Yes	NO	NO	NO	myset = {1, 2, 3}
Dictionaries	unordered collection of Key-value Pairs	Yes	Yes	NO	Yes	mydict = {'key1': 'value1', 'key2': 'value2'}

Examples of Lists:

→ Lists are ordered mutable sequences that can be changed after they have been created by adding removing and changing objects.

prices = [20, 40, 10, 5]

→ A ~~tuple~~ is created and stored data in ~~tuple~~ list form.

ages = [15, 18, 19, 25]
print(ages)

→ print will give (or) show the data (or) element's in ages and print it has output.

fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)

→ First a list is created has fruits
→ using the "append()" function we add the new element (or) data in the list
→ The orange is added at the last of the list.

output = ["apple", "banana", "cherry", "orange"]

Examples of tuples:-

→ Tuples are ordered immutable sequence that store multiple items.

```
Fruit's = ("apple", "banana", "cherry")
```

→ single object tuples are referred to as a singleton.

→ It can be created by using a trailing comma after the object, or else python identifies it as a string.

```
Fruit = ("orange")
type (Fruit)
str
Fruit = ("orange",)
```

→ commas are what makes a tuple, as parentheses are optional.

```
letters = 'a', 'b', 'd'
type (letters)
tuple
```

→ A tuple can be created by a pair of parenthesis and common-seperated objects.

```
mytuple = (1, 2, 3)
print (my_tuple)
```

→ tuple's are respresented by "()"

→ Here we create a new tuple

→ print is used to get the output.

Examples of sets:-

→ set can be created by using curly brackets following the variable names (or) using set() constructor.

```
letters = {'b', 'a', 't', 'i'}

letters = set(('b', 'o', 't', 'i'}
```

→ set are used to store multiple items in a single variable.

Creating a set.

```
Fruits = {"apple", "banana", "cherry"}

print (Fruits)
```

→ Here we create a new set like Fruits
→ Print is used to get output

Join two set

```
S1 = {"a", "b", "c"}
S2 = {1, 2, 3}

S3 = S1.union(set2)
Print(set3)
```

→ S_1 and S_2 will be created has set
→ S_1.union() using we combine the two set and store in set S_3

output will be {'c', 1, 'b', 2, 3, 'a'}

→ Print is used to get output

Examples of Dictionaries:

→ the values associated with a key can be any object. like sets, dictionaries are unordered.

```
age = {"mike":10,
       "Leo":19,
       "Dea":5
      }
print(age)
```

output is {'mike':10,'Leo':19, 'Dea':5}

→ values in dictionary items can be of any data types.

```
mydict={'key1':'value1','key2':'value2'}
print(mydict)
print(mydict['key1'])
mydict['key3'] = 'value3'
print(mydict)
```

→ Print will be used to get output.

Dictionary items:

→ Dictionary items are presented in key: value pairs can be referred by using the key name.

```
thisdict ={
    "brand": "ford"
    "model": "mustang"
    "year": 1964
}
print(thisdict["brand"])
```

→ Here we create the new dictionaries with each special and it's name

→ Print represent brand which will print name of Brand "output is ford"