Tuples

- Tuples are used to hold together multiple objects. Think of them as similar to lists, but without the extensive functionality that the list class gives you. One major feature of tuples is that they are immutable like strings i.e. you cannot modify tuples.

- Tuples are defined by specifying items separated by commas within an optional pair of parentheses.

- Tuples are usually used in cases where a statement or a user-defined function can safely assume that the collection of values i.e. the tuple of values used will **not change**.

```
# The intern at NASA

planets = ["mercury", "venus", "earth", "mars", "jupiter", "saturn",
"neptune"]

planets[2] = "Rahul"

planets

['mercury', 'venus', 'Rahul', 'mars', 'jupiter', 'saturn', 'neptune']
```

```
# Immutable

t = ("mercury", "venus", "earth", "mars", "jupiter", "saturn",
"neptune")

type(t)

tuple

t[2] = "Rahul"

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
/var/folders/zn/hkv6562d6_d30glfs8yc76900000gn/T/ipykernel_4277/145881
9811.py in <module>
----> 1 t[2] = "Rahul"

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

```
# Making an empty tuple
# 1st way
t1 = ("rahul",)
type(t1)
tuple


# 2nd way
t3 = ()
type(t3)
tuple

# 3rd way
# Quiz
t2 = tuple()
type(t2)
tuple
t4 = tuple("Rahul")
t4
('R', 'a', 'h', 'u', 'l')
type(t4)
tuple
tuple('hello world')
('h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd')
tuple([1, 2, 3, 4])
(1, 2, 3, 4)
```

```
# Iteration, indexing and slicing
```

```
t5 = (3, 4, 6, 7, 3)
```

```
len(t5)
```

```
5
```

```
for i in t5:
    print(i, end=" ")
```

```
3 4 6 7 3
```

```
t5[::-1]
```

```
(3, 7, 6, 4, 3)
```

```
t5[1:5]
```

```
(4, 6, 7, 3)
```

```
# Quiz
```

```
print(type((1,2,3)))
```

```
<class 'tuple'>
```

```
# Count and index
```

```
# t.append("Pluto")
```

```
t
```

```
('mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'neptune')
```

```
t.count('mercury')
```

```
1
```

```
t.count(241)
```

```
0
```

```
t.index('earth')
```

```
2
```

```
# dir(t)
```

Sets

Problem: Suppose that you are working as a data analyst at an edtech company. The edtech company is offering courses on Calculus(C) and Linear Algebra(L) among many others. Now you want to represent the students in the courses. Which data structure to use?

- Assume all names are unique for the students taking part in the edtech company. We have done this for better understanding. We could have taken id which will be unique.

Options?

- Lists => It can contain duplicate values
- Tuples => It can also contain duplicate values.

There is a different alternative => Sets

```
admissions = {"Rahul", "Akash", "Monika", "Harshita", "Vinamra",
"Rahul", "Akash"}
```

```
print(admissions)
```

```
{'Rahul', 'Vinamra', 'Akash', 'Monika', 'Harshita'}
```

```
type(admissions)
```

```
set
```

```
# in operator
'Rohit' in admissions
```

```
False
```

Creation

- set()
- set(iterable)

```
s1 = set()
```

```
type(s1)
```

```
set
```

```
# Quiz
```

```
set('hello world')
```

```
{' ', 'd', 'e', 'h', 'l', 'o', 'r', 'w'}
```

Iteration and indexing?

- We cannot access a set item by referring to an index or a key. If you cannot access an element we cannot modify it. That's why it is also unchangeable
- We can however run a loop to iterate.

```
admissions
```

```
{'Akash', 'Harshita', 'Monika', 'Rahul', 'Vinamra'}
```

```
for i in admissions:
    print(i)
```

```
Rahul
Vinamra
Akash
Monika
Harshita
```

```
# Challenge: Count number of unique elements in a sentence
```

```
sent = "be the change you wish to see in the world"
```

```
l = sent.split()
print(l)
```

```
['be', 'the', 'change', 'you', 'wish', 'to', 'see', 'in', 'the',
'world']
```

```
len(sent.split())
```

```
10
```

```
t = set(l)
```

```
print(t)
```

```
{'be', 'in', 'you', 'wish', 'see', 'change', 'the', 'world', 'to'}

len(t)

9
```

Changing a set

- add: For single element
- update(iterable)

```
# add

# Quiz

# s = set()
# s.append(5)

tour = {"Blanket", "Clothes", "torch", "toilet kit"}

tour.add("charger")

print(tour)

{'charger', 'Clothes', 'toilet kit', 'Blanket', 'torch'}

tour.add("tent")

tour

{'Blanket', 'Clothes', 'charger', 'tent', 'toilet kit', 'torch'}


# update

toilet_kit = ["soap", "face wash", "brush", "totthpaste", "perfume"]

# tour.update(iterable)

tour.update(toilet_kit)

tour

{'Blanket',
 'Clothes',
 'brush',
 'charger',
 'face wash',
 'perfume',
 'soap',
 'tent',
 'toilet kit',
 'torch',
 'totthpaste'}
```

Deleting an element

- pop: removes random element. We are not sure what it is
- remove(element): Removes particular element

```
tour
```

```
{'Blanket',
 'Clothes',
 'brush',
 'charger',
 'face wash',
 'perfume',
 'soap',
 'tent',
 'toilet kit',
 'torch',
 'totthpaste'}
```

```
tour.pop()
```

```
'totthpaste'
```

```
tour.remove("toilet kit")
```

```
tour
```

```
{'Blanket',
 'Clothes',
 'brush',
 'charger',
 'face wash',
 'perfume',
 'tent',
 'torch',
 'totthpaste'}
```

```
tour.remove("torch")
```

```
tour
```

```
{'Blanket', 'Clothes', 'brush', 'charger', 'face wash', 'perfume',
'tent'}
```

Intersection

- Suppose you want to find out which students are enrolled in both the Calculus and Linear Algebra Course. Then you can use the intersection method.

```
# Common in both sets

calculus = {"Rahul", "Afifa", "Renuka"}
linear = {"Akash", "Amol", "Rahul", "Afifa"}

calculus.intersection(linear)

{'Afifa', 'Rahul'}
```

Union

- Suppose you want to find out which students are enrolled in either the Calculus or the Linear Algebra Course or in both. Then you can use the union method.

```
# All elements in both sets

print(linear)
print(calculus)

{'Rahul', 'Afifa', 'Amol', 'Akash'}
{'Rahul', 'Afifa', 'Renuka'}

linear.union(calculus)

{'Afifa', 'Akash', 'Amol', 'Rahul', 'Renuka'}
```

```
# Quiz

l = [1,1,2,2,3,3]
s = set(l)
print(len(s), s)

3 {1, 2, 3}
```

Difference

- Suppose you want to find out the set of students who have enrolled in the Calculus course but not in Linear Algebra course or vice-versa, then we can use the difference method.

```
# Elements present in a set but not in another. A - B

# quiz

calculus
```

```
{'Afifa', 'Rahul', 'Renuka'}
```

```
linear
```

```
{'Afifa', 'Akash', 'Amol', 'Rahul'}
```

```
calculus.difference(linear)
```

```
{'Renuka'}
```

```
print(calculus - linear) # same as calculus.difference(linear)
```

```
{'Renuka'}
```

```
print(linear - calculus)
```

```
{'Amol', 'Akash'}
```

```
# Quizzes
```

```
a = {1,2,3}
b = {3,4,5}
print(a-b)
print(a.union(b))
print(a.intersection(b))
```

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

```
set1 = {1, 2, 3, 4, 5, 6}
set2 = {2, 4, 5, 6, 7}
x = set1 - set2
print(x)
```

```
{1, 3}
```

```
# Doubts
```

```
a ="hello"
print(a[-5:0:])
```

```
t = ([12, 34])
```